1991-09

An examination of target tracking in the Antisubmarine Warfare System Evaluation Tool (ASSET).

Vebber, Paul W.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN EXAMINATION OF TARGET TRACKING IN THE ANTISUBMARINE WARFARE SYSTEMS EVALUATION TOOL (ASSET)

by

Paul W. Vebber

September 1991

Thesis Advisor: James N. Eagle

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 3A	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code)	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
AN EXAMINATION OF TARGET TRACKING IN THE ANTISUBMARINE WARFARE SYSTEMS EVALUATION TOOL (ASSET)

12 PERSONAL AUTHOR(S)
VEBBER, Paul W.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1991, September, 30	130

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Kalman Filter; Maneuvering Target Statistical Tracker; Integrated Ornstein-Uhlenbeck Process

19 ABSTRACT (Continue on reverse if necessary and identify by block number) The role of the Maneuvering Target Statistical Tracker (MTST), a Kalman filter tracking algorithm based on the Integrated Ornstein-Uhlenbeck (IOU) motion process, in the Antisubmarine Warfare Systems Evaluation Tool (ASSET) is a campaign simulation which models open-ocean ASW scenarios featuring prosecution of hostile submarines by friendly submarines and aircraft based on cues provided by data fusion centers. The heart of each data fusion center is an MTST which integrates new contact information into tracks. Comparing the level of sophistication of the tracking algorithm to that of the contact data provided to it, a number of simplifications are proposed. These include using reduced complexity IOU predicton and Kalman filter equations; the use of pre-processed variance data together with the true position of targets to estimate, rather than explicitly calculate, updated track states; and limiting contact processing based on information content. Results indicate a good simulation of tracker output is produced using a greatly simplified algorithm. This technique can be generalized to other types of simulations involving target tracking.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL EAGLE, James N.	22b TELEPHONE (Include Area Code) 408-646-2214 22c OFFICE SYMBOL 3A

DD Form 1473, JUN 86 *Previous editions are obsolete.* S/N 0102-LF-014-6603

Approved for public release; distribution is unlimited.

An Examination of Target Tracking in the Antisubmarine Warfare System Evaluation Tool (ASSET)

by

Paul W. Vebber
Lieutenant, United States Navy
B.S., University of Wisconsin-Madison , 1984

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED SCIENCE

from the

ABSTRACT

The role of the Maneuvering Target Statistical Tracker (MTST), a Kalman filter tracking algorithm based on the Integrated Ornstein-Uhlenbeck (IOU) motion process, in the Antisubmarine Warfare System Evaluation Tool (ASSET) is examined and its operation described. ASSET is a campaign simulation which models open-ocean ASW scenarios featuring prosecution of hostile submarines by friendly submarines and aircraft based on cues provided by data fusion centers. The heart of each data fusion center is an MTST which integrates new contact information into tracks. Comparing the level of sophistication of the tracking algorithm to that of the contact data provided to it, a number of simplifications are proposed. These include using reduced complexity IOU prediction and Kalman filter equations; the use of pre-processed variance data together with the true position of targets to estimate, rather than explicitly calculate, updated track states; and limiting contact processing based on information content. Results indicate a good simulation of tracker output is produced using a greatly simplified algorithm. This technique can be generalized to other types of simulations involving target tracking.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I wish to express my sincerest thanks to Professor Jim Eagle for always providing encouraging guidance and supportive critique as I went from confusion, to wonder, to understanding and back to confusion again, innumerable times during my struggle with the workings of ASSET, Kalman filters, and the Integrated Ornstein-Uhlenbeck process. His reassurance that there was indeed light at the end of the thesis tunnel helped make my thesis experience a positive, though stressful, part of my studies. A note of thanks goes to all my friends in ASW class IX01, especially Tom and the Red Sox, who ensured I kept a proper perspective on things.

I arrived here alone, but I leave with the love and companionship of the most wonderful person and best friend I have ever met, my wife, Karen. Her supreme confidence in my ability to succeed and her understanding acceptance of late nights and boring weekends, allowed me to work those hours knowing a glowing smile and a warm embrace would meet me when I quit for the night. You've earned quite a honeymoon! Thank you with all my heart, I love you! We can go home now.

I. AN INTRODUCTION TO ASSET

The Antisubmarine Warfare Systems Evaluation Tool (ASSET) is a campaign-level simulation which models open ocean scenarios involving submarines, maritime patrol aircraft (MPA), shore-based command and data-fusion centers and a wide variety of passive acoustic sensors. Active acoustic and non-acoustic sensors are only modeled using a simple area sensor model which has a fixed probability of detection and false alarm rate. The simulation scenario is specified by a user-supplied "architecture" which determines all facets of environment, command control, sensor interaction and platform maneuver. This structure is input to a desktop computer/workstation (the Apple Macintosh II in this version) through a series of user-friendly windows, each dealing with a specific topic. A particular configuration of sensors, data fusion centers, communication nodes and tactical platforms interact in a particular geographic location against an analogous enemy force structure.

The scenario is repeated as a Monte Carlo simulation to produce statistically meaningful measures of effectiveness (MOE's). Output data regarding the detection, localization and prosecution of enemy submarines can then provide a quantitative basis for decisions regarding ASW Master Plans, Top Level Warfare Requirements, and a variety of emerging technology assessments and system appraisals [Ref. 1:p. iv]. These results can also be useful in conducting quantitative experimentation with regional force

levels, force compositions and commitment strategies; command, control, communications and intelligence (C^3I) networks; implications of foreign technology advances; and fleet exercise planning. The current version of ASSET (version 1.0) limits the scope of these scenarios to open ocean search and prosecution of hostile submarines by cueing friendly submarines and MPA with wide-area sensors, and modeling the supporting C^3I networks. The modular nature of the object-oriented structure of ASSET makes expanding the scope of the simulation possible [Ref. 2:p. 1-2]. As the construction and operating costs of naval platforms spiral upwards, a simulation tool of this kind will be increasingly important to intelligently manage resources vis-a-vis a rapidly evolving spectrum of possible threats.

A. OBJECTIVES AND RATIONALE

The purpose of this thesis is three-fold:

- to provide a description of the target tracking algorithm used by ASSET and how it relates to:
    1. real-world tracking.
    2. the ASSET simulation as a whole.
- to examine the mathematical development of the tracking algorithm.
- to examine possible modifications to the existing Kalman filter tracking algorithm to better match the data input to it and increase computational efficiency.

The use of full Kalman filter equations for elliptical areas of uncertainty (AOU's) is not necessary given the circular AOU's that ASSET's sensor model produces. Making use of the fact that the computer knows the ground truth locations of the platforms, and the limited data input to the filter, it will be shown that it is possible to approximate appropriately distributed filtered state positions using simplified forms of the Kalman filter matrix computations. The situations when the tracker algorithm is used can also be limited based on the information content of the contact. Together, these modifications yield a significant reduction in the computational complexity of the tracking algorithm, which is the principle goal of this thesis.

The automatic correlator tracker (ACT) implemented in ASSET is a stand-alone module based on the Ocean Surveillance Information System (OSIS) baseline upgrade single hypothesis, multiple target, Kalman filter-based, correlation and tracking algorithm [Ref. 3:p. 4]. This set of routines contains a complete representation of the OSIS baseline upgrade automatic correlator tracker's (OBU-ACT) functions, with the exception of LINK, the utility that evaluates track sets which are potentially legs of a single target's track; and EQUATE, which makes track associations for contacts pre-correlated to a particular platform based on acoustic or electromagnetic signature.

The OBU-ACT package is designed to process contact information derived from position-only, bearing-only, and position and velocity reports. The covariance matrices associated with these reports can be interpreted as an elliptical representation of the error in position and velocity. The capability to process line of bearing

contacts is not available in ASSET (1.0). The method used to model sensor systems in ASSET 1.0 approximates the elliptical errors with circular ones. This reduces the complexity of the covariance matrices considerably. Taking advantage of the large number of zero elements and repeated values in the matrices actually constructed by ASSET, the matrix equations involved in the tracking algorithm can be reduced to equivalent scalar ones by eliminating all the zero-multiplied terms.

Central to this algorithm is the Integrated Ornstein-Uhlenbeck (IOU) process which models target motion in the Maneuvering Target Statistical Tracker (MTST). The MTST uses this model, together with contact data corrupted by noise, to estimate the size and position of the Submarine Probability Area (SPA) which has an 86 percent probability of containing the target's true position. The major modification to the algorithm proposed is based on assuming that the IOU prediction position distribution, in like fashion to the contact position distribution, is centered on the target's true position. This results in a distribution of SPA centers that is centered on the targets true position also. Thus, only the SPA variance is calculated, not its center position, which is drawn randomly from the estimated distribution about the true target position. Since contact data does not play a role in computing these variances, they may be calculated prior to the start of the simulation. A table of variances for the sensor AOU values defined for the scenario and an appropriate range of contact report interarrival times can be constructed and referred to as needed during simulation execution to minimize the time spent processing contact data.

B. FUNDAMENTALS OF OPERATION.

The basic structure of ASSET was adapted from COAST, the Common LISP Architectural Study Tool. Common LISP is an object-oriented programming language used for rapid prototyping and artificial intelligence work. A general overview of this structure is presented to facilitate understanding of how the COAST/ASSET system is organized and the critical role of the tracker to simulation operation.

1. Objects and Object Interactions.

Object-oriented programs combine data with associated procedures to form a hierarchical network of self-contained modules known as objects. These objects can then be invoked by each other according to the program methodology. Program objects are independent and can be modified without affecting other objects they communicate with. The data that is communicated will change, but the relationship between the objects does not necessarily have to. Data can be evaluated internally by an object without affecting the analogous data in a separate object.

The principle of *instancing* allows a single object definition to create any number of structures within the program, all obeying the same definition. This allows a single definition of a submarine object, for example, to be given many different sets of parameters each representing a different class of submarine. Other objects may be designated to impart their characteristics to the new submarine object. This process of *inheritance* allows the properties of another object, an acoustic sensor object for example, to be created separately from the submarine object. The

submarine object can then be designated to be a kind of acoustic sensor and it will inherit the properties of the acoustic sensor object in addition to its own. Another object such as a SURTASS towed-array surveillance ship object can also inherit the properties of the acoustic sensor object, with different parameters, without affecting the parameters of the acoustic sensor inherited by the submarine object.

Complex classes of platforms can be formed by multiple inheritance of the properties of basic building block objects. These are then used to simulate any number of platforms of that class, each operating independently within the simulation. The computer memory available to store each of these instances of the object becomes the limiting factor to the complexity of the scenario to be examined. The organization of objects in ASSET consist of several major groups:

- Graphical environment objects which implement the standard Macintosh graphical user interface to provide input/output by means of a mouse based "point and click" metaphor featuring pull-down menus,dialog boxes with labeled data regions and radio buttons, and windowed presentation of multiple information sources simultaneously.

- Region management objects which construct and manage distinct regions of varying environmental properties, command responsibility or platform patrol assignment.

- Event management objects which queue all simulation events in time order sequence and parcel them out to the appropriate resolution objects.

- Command objects which perform resource allocation, assigning available assets to contact cues based on either time to station (MPA) or area of uncertainty size (submarines).

- Automatic Correlator Tracker (ACT) objects which manage the grouping and processing of contact reports into target tracks creating a tactical picture consisting of combinations of true and false contacts.

- Tactical platform objects which simulate the behavior of their real world counterparts. These are dynamic associations of component objects which may change in response to simulation events. A submarine inheriting the properties of a Patroller object may switch to an Intercepter after making a detection.

The program flow from object to object is not sequential, but event driven. Once started, the simulation clock advances time relative to the simulation. Sensor objects begin glimpsing and motion platforms begin moving. Possible detection events are sent to the event manager for proper sequencing and resolution. Detections are communicated up the designated chain of command to data-fusion centers where an instance of the ACT processes them and integrates them into the tactical picture. This may cause a command object to direct (or redirect) assets in response. This process continues until the allotted time for the scenario expires. The MOE statistics for that run are added to the MOE file and the next iteration of the scenario commences. When the desired number of iterations are complete the compiled summary of MOE statistics is analyzed to interpret the results of the simulation.

2. ASW System Architecture Evaluation.

In order to evaluate a desired architecture it must be conceptualized in great detail. The classic computer maxim "garbage in, garbage out" is especially true of ASSET where a single bad parameter value can render the results meaningless. While the user-friendly interface facilitates the mechanical process of inputing data, the abstract nature of the required parameters make it necessary to assume values of dubious validity at times. General areas which require quantified data include:

- Communication connectivities which represent how detection reports are passed from object to object and what delays are involved at each node.

- Command organization including geographic regions of responsibility.

- Environmental data for acoustic propagation loss as a function of range and frequency as well as ambient acoustic noise in each distinct environmental region.

- Motion plans representing the operating areas and interconnecting tracks which will govern tactical platform movement.

- Umpire parameters such as the kill probabilities for a given submarine class against each possible target class both for the case where it detects the enemy first and the case where it is detected by the enemy first.

The individual platforms require complex definition as well. A submarine object, for example, requires parameters for:

- breakoff speed between "fast" and "slow" acoustic behavior.

- self-noise at slow and fast speed.

- directivity index and recognition differential (for Sonar Equation computations).

- signature frequency emitted and intensity at slow and fast speed.

- movement speeds when patrolling and intercepting and detailed motion plan to be followed.

- weapons loadout and level at which the sub will abort its mission to rearm.

- whether the sub will transmit the detections it makes and risk detection itself or not report any detections.

- the interval at which it copies the submarine broadcast for orders to investigate cues. [Ref. 2:p. 2-37]

The submarine's sensor requires a detailed set of parameters of its own which will be used to construct the contact report that is communicated to the ACT representing the data fusion center associated with the submarine's command object:

- position error which represents the 2 standard deviation radius of a circular normal distribution of contact errors about the actual target position.
- course and speed errors which are uniform about the true value.
- Target Motion Analysis (TMA) delay representing the time required to acquire course and speed information.
- false alarm rate.
- determination whether, on the average the sensor is reliable enough for the fusion center to automatically start a new track based on a single contact report.

The complexity of the data involved in constructing an architecture for evaluation varies considerably, as can be seen from these examples. Once all this information is input, it can be easily double checked by accessing the appropriate windows again. When the user is sure all data is input correctly, the simulation can be run with the platform graphics turned on to ensure it progresses properly. These graphics can then be turned off to run multiple iterations faster.

When the desired number of iterations is completed a dialog box opens asking if the MOE's are to be saved. Once saved to disk, they can be opened and examined. They include:

- Attrition of submarines and MPA assets for each side as a fraction of original force.
- Number of times submarines approached to within a critical range (assigned by the user) of enemy surface formations (which serve only as targets in ASSET 1.0, they have no combat or detection capability).
- Tracker statistics involving the fusion delay involved in constructing target tracks.

These statistics along with weapon expenditures allow the completed simulation to be compared to other runs to observe the effect of a particular parameter changing, or how close a particular MOE comes to a goal value. The methodology of ASSET must be taken into account as the computer commanders follow simple resource allocation rules which do not necessarily reflect the tactical priorities the user desires. A higher priority for prosecution may well be given to a submarine returning to base than one ten miles from a surface group, due to the resource allocation process' emphasis on detection rather than on protection. While creativity is needed in the design, implementation and interpretation of an ASSET scenario, significant insights into the conduct of ASW campaigns can be gleaned from it.

3. Correlation, Tracking and Data Fusion.

ASSET is designed to simulate the data fusion centers where contact reports from a wide variety of sensors and platforms are centrally processed to create a tactical picture for the region of interest. These data fusion centers are simulated by an instance of the OBU-ACT module with appropriate communication connectivities. The result of correlating and processing the streams of contact information arriving at

a fusion center forms the basis for the allocation and cuing of assets. Thus it is critical that the ACT is not spinning its wheels performing unnecessary computations.

The implementation of OBU-ACT used in ASSET is organized into four main functional areas:

- generating a new track from one or more contact reports (the START and CLUSTER modules).
- associating a contact report to an existing track (the INPUT, START, and ASSOC modules).
- simulating the intervention of a human analyst to resolve ambiguous contact-to-track associations (the ANALYST module).
- updating and managing a database of the status of all contact reports and tracks (the Locational Data Base Manager and MTST modules).

The data flow through these modules is shown in Figure 1. A detection is made by a sensor and a report is initiated. After the applicable delays have elapsed, the contact report travels along the communications network to the fusion center, experiencing additional delay at each node.

Upon arrival at the fusion center the INPUT module enters it into the Locational Data Base Manager (LDBM) module and passes it to the ASSOC module. There the contact is compared to the existing tracks on the basis of geofeasibility. Those tracks with which the contact could be associated are evaluated by a statistical comparison of a measure of correlation (MOC) with a preset threshold. If the contact fails to trigger an association with any track it is passed to the START module. If the contact triggers a single association, or if one track MOC exceeds all others by a preset

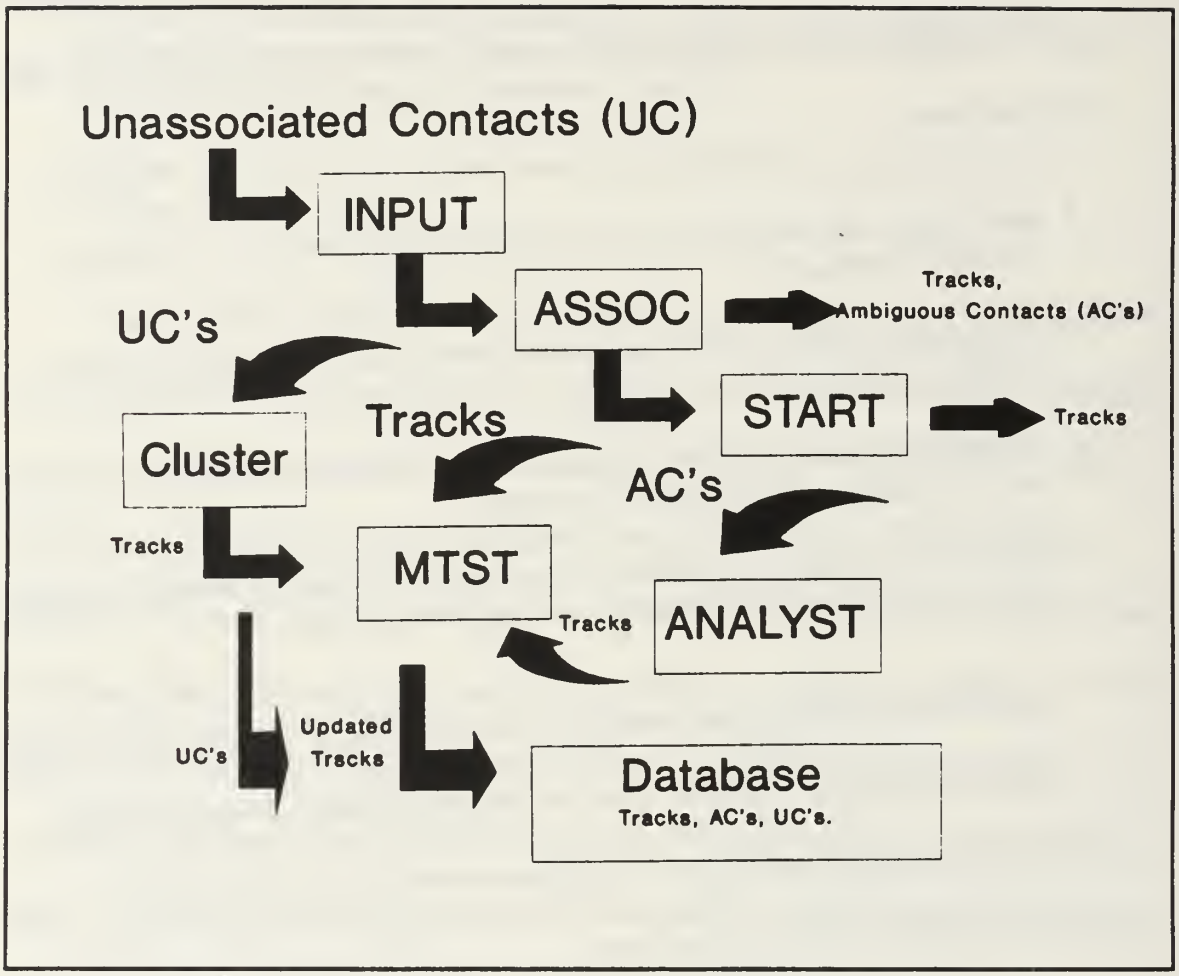


Figure 2: ASSET contact data flow.

margin, it is unambiguously associated with that track [Ref. 4:p. 3]. Ambiguous association is resolved by the ANALYST module, which has a given probability of making the correct association (.7) and an associated exponentially distributed delay time spent making the decision (mean of .2 hours) [Ref. 3:p. 11; Ref.5:ACT-ANALYST].

The contacts which are unresolved are passed to START which checks the value of the flag **single-report-to-track**. If the value is true, START initiates a new track based on the contact, if it is false, the contact is passed to the multiple contact track initiation module CLUSTER. This represents a simplification of the actual OBU-ACT START module which uses comparison to a set of criteria and human analyst interaction to determine if a new track is to initiated [Ref. 6:p. 3-4]. Within the context of the simulation, this simplification makes sense as two classes of detecting sensors are represented; those that report continuous contact observations such as SOSUS or other fixed area sensors, which are processed at the fusion center; and those which perform platform-level analysis and periodically report the tracks they hold, such as submarines. Platforms in the first category, unless possessing unusually low false alarm rates, would not initiate a track based on a single report, while those in the second, would.

Unassociated contacts which remain after passing through the ASSOC and START modules are processed by the CLUSTER module. Here subsets of all unassociated contacts are evaluated for geofeasibility and constant course and speed likelihood. If the value of the constant course and speed likelihood exceeds a threshold value then the subset of unassociated contacts is used to initiate a new track [Ref. 7:p. 1-3]. Contacts which are not associated after passing through ASSOC and CLUSTER carry counters which increment each time they pass through the two modules. When these counts exceed maximum limits (hardwired at 7 in CLUSTER and at 5 in ASSOC) they are dropped from the LDBM [Ref. 5: OBU-ACT].

When a new track is initiated or a contact is associated with a track, the MTST module is called to filter the track data and produce an optimal estimate of the target's location and velocity along with the error covariance matrix describing the quality of the estimate. The full capability tracker in MTST contains the matrix versions of the Kalman filter equations. These equations determine the mean target position at a future point in time and the covariance matrix representing the error in that prediction. The elements of this covariance matrix can be recast as ellipses representing two σ or 86 percent containment regions for the target's possible position and velocity. A complete description of MTST is the subject of Chapter II.

The foregoing process of correlating contacts to tracks and filtering those tracks to provide an estimate of target position occurs at each instance of a data fusion center object. When formulating an architecture, the fusion center's role must be carefully designed. Since no contact reports flow out of a fusion center, it is not possible to directly fuse the pictures at two or more fusion sites into a higher echelon center. The individual sensors originating the detection reports must send duplicate reports to all fusion centers which will be using the report in determining a tactical picture. The fusion center also keeps separate instances of the tracker to process surface contacts and subsurface contacts. Together with the START processing of single contact to track decisions, this assumes that a great deal of contact processing is going on below the level of the fusion center.

Assuming that reports arriving at the fusion center are pre-processed justifies several assumptions that the DETECTION-REPORTER object makes about the

reports it generates. Most important to streamlining filter operation is the assumption that all submarine probability areas (SPA's) are circular. While this represents a considerable inaccuracy for single contact line-of-bearing detections, if one assumes the contact report represents the accumulation of a significant number detections, the circular error assumption is more reasonable. Several types of detections are resolved in ASSET without explicitly modelling the geometry of the searcher and target at the time of detection (MPA and Fixed Area Sensors specifically). The calculation of the size and orientation of explicit AOU ellipses where possible would slow execution and add considerable complexity to the program, requiring even more specific user input sensor parameters.

The assumption that this circular AOU size is independent of range is more difficult to defend, in light of an expected linear relationship between the standard deviation of the position error and range. That assumption on ASSET's part can be remedied easily, for the cases where range is explicitly determined, by expanding the radius of a sensor's AOU linearly with range based on a user input bearing error. This would make the process of tabulating preprocessed variance data impossible, however, as the sensor AOUs would no longer be constant. A good compromise involves defining three AOU sizes for contacts at short, medium and convergence zone ranges. This would allow for some representation of the range effect on AOU size, with acceptable complication of the variance pre-calculation procedure.

The other tracking assumption ASSET makes is that the covariance of the velocity can be represented by the steady-state stationary cross-covariance of the IOU

process, independent of the sensor characteristics. This assumption is used to initialize the covariance matrix of a new track. This method is correct for a position-only contact which gives no information about velocity. Given position and velocity information however, this is only accurate if the sensor is so inaccurate that the IOU process velocity variance is small compared to the sensor velocity variance [Ref. 8:p. 5-20].

This is not true in most cases however, with the result being that the filter re-initializes the velocity update each time a contact is processed, never including any representation of the velocity accuracy of the sensor involved. Thus, the velocity variance is based solely on the IOU model and not on the variance of the velocity reported by the sensors. This assumption makes the tracker more responsive to course changes, but it also artificially inflates the SPA size of accurate contact streams because it ignores the sensors velocity error, which may be significantly less than the IOU value.

Given these assumptions and the noise characteristics of the IOU process, the filter implementation can be streamlined to better match the data provided. This process is detailed in the following chapters which will describe the IOU process and the formulation of the general and ASSET-specific filter equations.

II. THE MANEUVERING TARGET STATISTICAL TRACKER

Maneuvering Target Statistical Tracker (MTST) is the name given to a class of Kalman filter-based estimation and prediction algorithms utilizing the Integrated Ornstein-Uhlenbeck process to model the statistics of target motion. To successfully track a maneuvering target, the details of the target path should be statistically predictable. How accurately an automated tracker like MTST performs its task is directly related to how well the tracker's target motion model describes the targets actual random motion.

The random tour model [Ref. 9] can be used to describe the statistical properties of a target moving at constant speed which makes random course changes at times separated by intervals chosen from an exponential distribution. This provides a reasonable estimate of the type of motion expected of a patrolling submarine target. Unfortunately, the target distribution generated by a random tour is not normal and cannot be used directly with a Kalman filter [Ref. 8:p. 2-10]. A process is required which produces normal distributions which best approximate the mean and variance of the random tour. This is the IOU process. The mathematical development which follows is summarized from the more complete treatment in Reference 8, the results of which are in agreement with References 5, 10 and 13.

A. THE INTEGRATED ORNSTEIN-UHLENBECK PROCESS

The Integrated Ornstein-Uhlenbeck Process (IOU) is based on a first order stochastic linear differential equation which approximates the higher order non-linear deferential equation which defines the motion of a randomly maneuvering target. It is a member of a class of equations known as Langevin equations and has the form:

$$\frac{d}{dt}u(t) = -\beta\, u(t)dt + \sigma\, dw(t). \qquad (2.1)$$

The first term represents a deceleration caused by a resistive force proportional to the velocity $u(t)$. The random forces acting on the particle are represented by the second term, where $w(t)$ is a Gaussian white noise process.

The stochastic nature of this equation makes it possible to find only the statistics of the distribution of the solutions, rather than the specific solution itself. A Gaussian $w(t)$ produces a Gaussian velocity process $u(t)$ which, like any normal process, is defined completely by its first and second order moments. The result is the velocity distribution of a particle which is undergoing random motion similar to Brownian motion, experiencing random instantaneous accelerations, but whose velocity is damped by a spring-like effect which constantly accelerates the particle in the direction opposite its velocity at a rate proportional to that velocity. The position variance of such a particle is unbounded over time, but the velocity variance is limited by the damping coefficient β. The result is a velocity distribution that is normal with a limiting variance given by:

$$\lim_{t\to\infty} V\{u(t)\} = \lim_{t\to\infty} E\{[u(t)-E\{u(t)\}]^2\} = \frac{\sigma^2}{2\beta}, \qquad (2.2)$$

and a mean for any given t given by:

$$E\{u(t)\} = e^{-\beta t} E\{u(0)\}. \qquad (2.3)$$

The mean and variance expressions for the position of an object whose motion is governed by an IOU process completely specify its positional distribution over time. In order to show that the IOU process approximates the same motion as a random tour, the radial position distributions for the two stochastic processes will be shown to have the same variance.

B. EQUIVALENCE OF THE RANDOM TOUR AND IOU PROCESSES

The variance of the radial distance from $(x(0),y(0))$ is:

$$E\{R^2(t)\} = E\{[x(t)-x(0)]^2+[y(t)-y(0)]^2\}. \qquad (2.4)$$

For the Random Tour, the following holds:

$$E\{R^2(t)\} = \frac{2V}{\alpha^2}[\alpha t-1+e^{-\alpha t}], \qquad (2.5)$$

where V is the speed of the randomly touring particle and α is the mean number of course changes per unit time. The corresponding result for the IOU process is:

$$E\{R^2(t)\} = \frac{2\sigma^2}{\beta^3}[\beta t - 1 + e^{-\beta t}], \qquad (2.6)$$

where σ is the scale factor for the random acceleration and β is the damping constant on the velocity, as in Equation (2.1).

These two equations can be made identical given the following relationships hold:

$$\beta = \alpha \qquad \frac{\sigma^2}{\beta} = V^2 \qquad (2.7)$$

Thus, a Random Tour with parameters α and V can be approximated by an IOU process with parameters $\beta = \alpha$ and $\sigma = V\alpha^{1/2}$. It is worth noting that since a normal distribution is completely specified by its mean and variance, the IOU process represents the best normal approximation to the Random Tour [Ref. 10].

In ASSET the IOU parameters, β and σ are hardwired to represent a target conducting a random tour at ten knots with an average time between course changes of four hours. These parameters should match the actual motion of the platforms modelled in a given scenario. Since the user can choose whatever target motion parameters he wants for each individual platform, the parameters embedded in ASSET may disagree considerably with the actual target motion. This disagreement causes the IOU prediction to consistently lead or lag the target's mean position, depending on whether the model's speed and course change rate correspond to a velocity faster or slower than the target's speed, respectively. This causes the position of the center of the resulting SPA to be offset from the true target position.

Even when the tracker parameters exactly match the target motion, a considerable lag occurs if the time between measurements is not significantly greater than the time between maneuvers.

C. OTHER MOTION MODEL OPTIONS.

The IOU model used by MTST damps out the mean speed of the target exponentially over time to account for target maneuvers. No matter how small the time interval, the process reduces the mean speed by the proper amount. If the interval between detections is consistently shorter than the maneuver interval, the damped velocity is used to predict the position of the constant velocity target and the tracker develops a lag. One remedy for these inconsistencies would involve increasing the complexity of the tracker by using adaptive methods to more closely match the model parameters to a given target's track. An adaptive filter recognizes changes in the targets motion and compensates for it by changing the parameters of the motion model.

Desiring to reduce the filter's complexity however, instead of explicitly modelling an adaptive tracker, the use of such a tracker can be approximated using the IOU process as a basis for the size of the SPA, but assuming the model has some adaptive properties to position it more accurately. This can be accomplished using the existing IOU model to estimate the SPA variance and by assuming the tracker's position prediction's are distributed about the target's true position in a circular normal fashion. This is an optimistic assumption, but captures the fact that real world

trackers, using more complete contact data than ASSET and an adaptive motion model, can achieve more accurate position and velocity estimates. Examinations of such trackers can be found in References 14, 15 and 16. Reference 14 in particular demonstrates a passive tracker that delivers outstanding accuracy while the contact is on a constant course and speed, but has difficulty maintaining track during difficult target maneuvers.

The benefits of adaptive tracking are not applied to the SPA size in this case, as the ASSET tracker is not an adaptive one, but are assumed in simplifying how its position is determined. This technique introduces a moderate degree of error in SPA size by not precisely simulating the mechanics of distributing the position of the center of the SPA. It also does not take into account the filter's velocity components. Reference 15 provides methods for countering these effects through noise adaptation and correlated maneuver gating while Reference 16 uses bias-sensitive maneuver detection and Kalman gain adaptation. Another option used to improve the IOU model is the use of a dual velocity system which combines a short term IOU process combined with a long term one. This type of model can reproduce a variety of motions depending on the weighting factors given to the short and long term components. The resultant Kalman filter has six states and a 6 x 6 covariance matrix. This is the implementation of the IOU process currently used in the TOMAHAWK fire control system for surface ship motion modelling [Ref. 8]. These are but a few of the schemes more complex trackers can use to improve motion modelling. The single velocity IOU process, while not adaptive, comes close to satisfying the

necessary conditions and since it is the basis of the ASSET tracker will be used for comparison.

D. THE KALMAN FILTER

In order to estimate target position, a method of combining the contact position and its covariance with a prediction of that position and covariance based on the IOU process is necessary. The Kalman Filter provides such an estimate, provided the process model is linear and time invariant, which the IOU process is. The Kalman filter is derived from the least-squares estimation theory of Gauss and Legendre [Ref. 11:p. 7]. This technique sought the most probable value of an unknown quantity based on a series of measurements containing unknown errors. More formally, finding the most probable estimate of x based on the matrix equation:

$$z_k = H_k x + v_k \tag{2.8}$$

where vector z_k represents a measurement taken at a discrete time k, based on an observation matrix H revealing information about one or more state parameters of the true state vector x but corrupted by a noise vector v. This technique was generalized by Kalman to apply to linear filter theory, producing a filter which produced a best linear estimate of x [Ref. 12].

The resulting filter is based on two mechanisms operating sequentially, one predicting the state and covariance for a future time and the other combining this prediction with a noisy measurement of the target's state taken at that time. This

should, given correct motion modelling and noise parameters, produce a state and covariance which more accurately reflects the true target state than either the prediction or the measurement alone.

These mechanisms are represented by two systems of matrix equations. The variables involved are defined in Table 1. The prediction equations for the state x and covariance Σ are:

$$\textit{state prediction}: \quad \hat{x}_{k+1} = \phi x_k + \mu_w, \tag{2.9}$$

$$\textit{covariance prediction}: \quad \hat{\Sigma}_{k+1} = \phi \Sigma_k \phi^T + Q_k, \tag{2.10}$$

where ϕ represents the transition matrix from the dynamic system model, Q represents the variance inherent in the dynamic model and μ_w represents the mean of the noise term w from the dynamic model, which is assumed to be zero in ASSET. The equations for combining the predicted and measured states and variances require the computation of a Kalman gain matrix which provides the weighting factors for the combination. These equations which update the filter's estimates are:

$$\textit{Kalman Gain}: \quad K_{k+1} = \hat{\Sigma}_{k+1} H^T \left[H \hat{\Sigma}_{k+1} H^T + R_{k+1} \right]^{-1}, \tag{2.11}$$

$$\textit{state update}: \quad x_{k+1} = \hat{x}_{k+1} + K_{k+1} \left[z_{k+1} - \mu_v - H \hat{x}_{k+1} \right], \tag{2.12}$$

$$\textit{covariance update}: \quad \Sigma_{k+1} = \left[I - K_{k+1} H \right] \hat{\Sigma}_{k+1}, \tag{2.13}$$

where R represents the variance of the noise inherent in the measurements, Z is the measurement itself, μ_v is the mean of the noise inherent in the measurement (also

Table I: Kalman Filter Definition of Terms

Kalman Term:	General Form	ASSET Form
State vector at time *k*:	x_k n x 1	mean 4 x 1
State transition matrix, time *k* to time *k+1*:	ϕ n x n	phi 4 x 4
State process noise mean:	μ_w n x 1	assumed zero
State error covariance, at time *k*:	Σ_k n x n	var 4 x 4
State process error covariance at time *k*:	Q_k n x n	f 4 x 4
Observation matrix:	H m x n	I 4 x 4
Observation at time *k*:	Z_k m x 1	state 4 x 1
Observation error mean:	μ_v m x 1	assumed zero
Observation error covariance at time:	R_k m x n	covariance 4 x 4

assumed to be zero in ASSET) and I is the identity matrix of proper size. The state and covariance updates, assuming H (as ASSET does) is the appropriate identity matrix can be written as:

$$\textit{state update:}\quad x_{k+1} = \Sigma_{k+1}\left(\hat{\Sigma}_{k+1}^{-1}\hat{x}_{k+1} + R_{k+1}^{-1}Z_{k+1}\right). \tag{2.14}$$

$$\textit{covariance update:}\quad \Sigma_{k+1} = \left(\hat{\Sigma}_{k+1}^{-1} + R_{k+1}^{-1}\right)^{-1}, \tag{2.15}$$

where the inverse of R_{k+1} and Σ_{k+1} are the weighting matrices. Thus the inverse of the noise error variances are the weighting factors by which the prediction and

measurement are linearly combined. This is the form of the Kalman equations which ASSET actually uses. The advantage of this form is the ready understanding of the underlying mechanism of weighted linear combination. The disadvantage is the computational necessity of performing three matrix inversions to calculate the variance propagation. Regardless of the form of the equations, the ϕ, Q and R matrices must be developed to complete the computational form of the filter.

1. Development of the ϕ Matrix.

The matrix ϕ represents the transition matrix which governs the dynamics of the state between two discrete times. The following development summarizes the derivation of Reference 11, Section 5.2. The differential equations defining the IOU process and velocity can be combined to form a single matrix differential equation which takes the following form in the single coordinate direction x:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{u}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \sigma \end{bmatrix} w(t) \qquad (2.16)$$

$$\textit{more compactly:}\quad \dot{X}(t) = F(t)\,X(t) + G\,w(t) \qquad (2.17)$$

where $X(t)$represents the system state vector and the *dot* over a symbol indicates differentiation with respect to time. Given that $X(t)$ has the value $X(t_o)$ at time t_o then $X(t)$ at any future time $t > t_o$ is given by:

$$X(t) = \phi(t,t_o)\,X(t_o) + \int_{\tau=t_o}^{t} \phi(t,\tau)\,G\,w(\tau)\,d\tau \qquad (2.18)$$

where $\phi(t,t_o)$ represents the *transition matrix* between the states at time t_o and time t. Provided that $F(t)$ is constant in time and assuming that t_o is zero, the matrix ϕ can be shown [REF. 8:p. 5-3,5-5] to have the form:

$$\phi(t) = \begin{bmatrix} 1 & \frac{1}{\beta}(1-e^{-\beta t}) \\ 0 & e^{-\beta t} \end{bmatrix} \qquad (2.19)$$

This represents the transformation projecting a state into the future based on the IOU model. The velocity mean decays to a new value which is simply the old value multiplied by $e^{-\beta t}$. The position is moved a distance equal to the time $1/\beta$ multiplied by the change in velocity:

$$x(t)_{k+1} = x(t)_k + \frac{1}{\beta}(u(t)_k - u(t)_{k+1}), \qquad (2.20)$$

where:

$$u(t)_{k+1} = u(t)_k e^{-\beta t}. \qquad (2.21)$$

The form this takes when expanded to cover both coordinate directions is:

$$\phi(t) = \begin{bmatrix} 1 & 0 & \frac{1}{\beta}(1-e^{-\beta t}) & 0 \\ 0 & 1 & 0 & \frac{1}{\beta}(1-e^{-\beta t}) \\ 0 & 0 & e^{-\beta t} & 0 \\ 0 & 0 & 0 & e^{-\beta t} \end{bmatrix}, \qquad (2.22)$$

where t is time elapsed from the last measurement to the observation time of the current measurement. It is important to notice that the transition matrix depends only on the IOU parameter β and not on σ. Thus the IOU velocity damping coefficient and the elapsed time determine the dynamics of the mean state and variance transition. Appendix A contains graphs of the values $\phi(1\ 2)$, which will be called $\phi 2$, and $\phi(2)$,which will be called $\phi 3$, take over various time intervals. The noise magnitude coefficient σ is involved in the noise terms R and Q.

2. Development of the Q Matrix.

The Q matrix represents the noise introduced by the white Gaussian process $w(t)$. As discussed above, as the elements of this matrix get large the filter increasingly relies on the measurement data and ignores its predictions. As the elements get small the filter increasingly ignores the new contact data and the errors in its predictions compound until it diverges, completely losing track of where its target is. The definition of this matrix, the process noise error covariance, is:

$$Q = E\left[\{Z-E(Z)\}\{Z-E(Z)\}^T\right], \qquad (2.23)$$

which for the random process operating in (2.26) becomes:

$$Q = E\left\{ \left[\int_{\tau=t_o}^{t} \phi(t,\tau)\, G\, w(t)\, d\tau \right] \left[\int_{s=t_o}^{t} \phi(t,s)\, G\, w(s)\, ds \right]^T \right\}. \qquad (2.24)$$

Since the mean of the white Gaussian processes $w(t)$ and $w(s)$ are both zero and the variables of integration τ and s are equal, this reduces to the form:

$$Q = \int_{\tau=t_o}^{t} \phi(t,\tau)\, G\, G^T\, \phi^T(t,\tau)\, d\tau. \qquad (2.25)$$

Substituting the appropriate matrices G and ϕ from (2.25) and (2.29) respectively and performing the indicated matrix multiplication results in:

$$Q = \int_{t=\tau}^{t} \begin{bmatrix} \frac{\sigma^2}{\beta^2}(1-e^{-\beta(t-\tau)})^2 & \frac{\sigma^2}{\beta}(1-e^{-\beta(t-\tau)})e^{-\beta(t-\tau)} \\ \frac{\sigma^2}{\beta}(1-e^{-\beta(t-\tau)})e^{-\beta(t-\tau)} & \sigma^2 e^{-2\beta(t-\tau)} \end{bmatrix} d\tau \qquad (2.26)$$

for the single coordinate direction x. Assuming that $t_o=0$ and recognizing that q(1,2) is equal to q(2,1) results in three expressions to integrate to get the final value of the Q matrix in one direction:

$$Q = \begin{bmatrix} q11 & q12 \\ q21 & q22 \end{bmatrix} \quad where \quad q12 = q21 \qquad (2.27)$$

$$q11 = \frac{\sigma^2}{2\beta^2}\left[2t - \frac{1}{\beta}(3 - 4e^{-\beta t} + e^{-2\beta t}) \right] \qquad (2.28)$$

$$q12 = q21 = \frac{\sigma^2}{2\beta^2}\left(1 - 2e^{-\beta t} + e^{-2\beta t}\right) \tag{2.29}$$

$$q22 = \frac{\sigma^2}{2\beta}\left(1 - e^{-2\beta t}\right) \tag{2.30}$$

When projected into two coordinate directions the full representation of the Q matrix is:

$$Q = \begin{bmatrix} q11 & 0 & q12 & 0 \\ 0 & q11 & 0 & q12 \\ q12 & 0 & q22 & 0 \\ 0 & q12 & 0 & q22 \end{bmatrix} \tag{2.31}$$

Each element in this matrix is directly proportional to σ^2 and decreases with increasing β. Time has an inverse exponential relationship causing the noise to increase the longer the interval between measurements. At t equal to zero the noise terms are equal to zero and as t goes to infinity :

$$q11 \to \infty, \quad q12 \to \frac{\sigma^2}{2\beta^2}, \quad q22 \to \frac{\sigma^2}{\beta}. \tag{2.32}$$

As mentioned previously, the variance in position is unbounded, but the cross-covariance and velocity covariance are bounded. These values become important in developing the measurement noise covariance matrix R. Appendix A contains graphs of these values over various time intervals.

3. Development of the *R* Matrix and Filter Initialization.

The matrix *R* represents the covariance of the measurement noise and accompanies each state measurement reflecting its precision. This value may vary with time in the ASSET tracker as contact reports flow in from a variety of sensors. The inverse of this matrix is used as the weighting function for the contact report, the larger the elements of *R* the less effect the contact has on the filtered update. The *R* matrix also plays a role in initializing the filter.

When a target is first detected, there is no *t-1* state for the filter to work from in making its prediction. In general, a time t_o *R* matrix is preset to initiate the filter. This matrix would consist of the 2 x 2 positional error matrix which describes an estimate of the elliptical AOU about the contact position:

$$\frac{1}{4}\begin{bmatrix} (A^2\cos^2\theta) + (B^2\sin^2\theta) & (B^2 - A^2)\sin\theta\cos\theta \\ (B^2 - A^2)\sin\theta\cos\theta & (A^2\sin^2\theta) + (B^2\cos^2\theta) \end{bmatrix} \qquad (2.33)$$

For a position-only contact, the two lower right diagonal entries would be set equal to the stationary variance of the IOU process (2.2). This represents the maximum variance the velocity could possibly have based on the motion model and since no velocity information is coming in, the filter is relying solely on the motion model, making this a value a reasonable choice:

$$\begin{bmatrix} \begin{bmatrix} EQ.\ 2.33 \end{bmatrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & \begin{matrix} \frac{\sigma^2}{2\alpha} & 0 \\ 0 & \frac{\sigma^2}{2\alpha} \end{matrix} \end{bmatrix}. \tag{2.34}$$

If the contact is a position-and-velocity contact, the procedure is more complex. In like fashion to the initialization of the upper left 2 x 2 portion with the variance in the sensor (its elliptical AOU), so must the bottom right 2 x 2 be initialized by the velocity variance of the sensor. Since the IOU process is predicting the velocity distribution also, this variance of the measurement must be combined with the variance of the IOU process. The proper combination can be shown [Ref. 8:App. F] to be the same as the alternate form of the Kalman covariance update equation (2.14).

This method is not used by the tracker in ASSET. The ASSET tracker simply uses the initial contact's IOU velocity variance as the basis for a new track, without updating it. This original contact report is used as the *t-1* state and covariance when the next contact associated with the track comes in. The contact reports themselves also differ from the construction described above. The positional variance of the sensors are all circles in ASSET so the 2 x 2 matrix which occupies the upper left of the R matrix is actually the 2 x 2 in the upper left of:

$$\begin{bmatrix} \frac{r^2}{4} & 0 & 0 & 0 \\ 0 & \frac{r^2}{4} & 0 & 0 \\ 0 & 0 & \frac{\sigma^2}{2\alpha} & 0 \\ 0 & 0 & 0 & \frac{\sigma^2}{2\alpha} \end{bmatrix} \tag{2.35}$$

The velocity variance in the lower right hand 2 x 2 is correctly set for a position-only contact, but this is also used for a position-and-velocity contact. Thus the R matrix which ASSET uses never incorporates the error in velocity inherent in the sensor. Using these values essentially reinitializes the filter each time a measurement is made. This is desirable for position-only contacts as it prevents divergence of the velocity components of the state, however for position and velocity contacts, this can cause poor velocity accuracy as the tracker improperly weights the velocity components of the contact report. The velocity information of different sensors are given equal weight regardless of the relative size of the errors in the reports they make. The actual velocity variance of the sensor should be computed and used to set the value of the filtered velocity using the form of (2.15).

III. IMPLEMENTATION OF THE TRACKER IN ASSET

Having developed the components of the Kalman filter, the construction of the tracker implemented in ASSET can be discussed. The following is based on Reference 5, the ASSET source code for version 1.0. The ACT-MTST module consists of objects which perform the filter calculations described above, as well as objects which maintain the spatial relationships between the positions involved on a spherical earth.

There are several constants which are defined for use in all the module's objects. The first is **dtor** which is used for degree to radian conversion. Where trigonometric functions are indicated below, the actual code uses **dtor** to convert angles from degrees to radians, but in the interest of brevity, this will be excluded. Second is α, which is equivalent to β in the IOU process, and is set to .25. Third is σ which is the noise scale factor from the IOU process as described above and set to $\sqrt{50}$. This is equivalent to the square root of α velocity multiplied by the Random Tour velocity. Thus the tracker predicts the target position by assuming a random tour is being conducted at a speed of ten knots with a mean interval between course changes of four hours. This cannot be changed by the user.

Inputs to ACT-MTST are read from the data in ACT-LDBM, the contact database manager. If a new track is to be initiated, the next MTST object, **start-new-**

track sets the track head and track tail state and covariance matrices equal to the contact mean and covariance using **compute-mean-covariance**. The use of two sets of data, a head and a tail, to allow incorporation of out-of-sequence contact reports will be discussed below. The object **compute-mean-covariance** is also called by ACT-ASSOC as a basis for the spatial measure of correlation used to determine contact to track correlation.

Compute-mean-covariance creates two arrays, a four by four called **var**, which is the R matrix, and a four by one called **mean**, which is z_k:

$$mean = \begin{bmatrix} 0 \\ 0 \\ spd\cos(90-cse) \\ spd\sin(90-cse) \end{bmatrix}, \qquad (3.1)$$

and **var**, which has the same form as (2.34). The vector **mean** is centered at the latitude and longitude indicated in the contact data list and x and y velocity computed from the contact course and speed. This variance matrix however, as discussed above, is calculated as if the contact report consists of an ellipse with minor axis B, major axis A and orientation angle θ. All the spatial data fields that are generated by the Detection Reporter and Generic Sensor contain a single positional uncertainty and an orientation angle of zero. Modifying (3.1) to take advantage of this fact results in (2.35) where r is equal to the standard deviation of the sensor's target position estimates. This alternative form returns matrix element values identical to those computed by (3.1), given the actual form of the contact reports generated.

If a contact is associated with an existing track rather than used to start a new one, the **update-track** object is called by ACT-LDBM to perform the Kalman filter operations. The remaining objects in ACT-MTST are all components of the **update-track** object with the exception of the object **cov-to-ellipse,** which is called by various display and graphics objects to determine the orientation, semi-major and major axes of the ellipse representing a covariance matrix. Since, as shown above, all the ellipses are actually circles, this object is another anachronism of the mismatch between the capabilities of the tracker and the contact reports actually input to it. It can be simplified in a manner similar to **compute-mean covariance** above to return the radius of a circle instead of a major axis, minor axis and orientation of a ellipse.

Focusing once again on the **update-track** object, it first defines the spatial data parameters of the contact and the track the contact is used to update. ASSET uses four data structures to represent the data associated with contacts and tracks:

- **obu-contact:** consisting of contact id, receipt time, track association, sensor, categorization (track association flags "pinnedp" and "lockedp", and number of passes through ACT-CLUSTER and ACT-ASSOC), spatial data (the **obu-data-field** described below), altitude (surface or subsurface) and HFDFp (an optional parameter associated with HFDF contacts).

- **obu-data-field:** type (position-only or position-and-velocity), observation time, latitude, longitude, AOU orientation, AOU major axis, AOU minor axis, course, course uncertainty, speed and speed uncertainty.

- **obu-track:** track id, number of contacts associated to the track (maximum of five), state and covariance of the track head (most recent contact) and track tail (oldest online contact), head and tail spatial data (each an **obu-data-field**), head and tail contact id, and altitude (surface or subsurface).

- **obu-state-field:** reference latitude, reference longitude, state array (offset from reference position in miles N/S, offset from reference position in miles E/W, N/S velocity, E/W velocity) and covariance array (four by four array per (3.1)).

The three AOU variables can be eliminated from **obu-data-field** and replaced by position-uncertainty, the radius of the actual AOU circle reported. This will not affect the tracker's calculation and reduces the memory taken up by contact reports.

An **obu-track** consists of up to five contact reports, the most recent of which is the track head and the oldest is the track tail. The variables required for performing the filter calculations are read from the data structures described above, **update-track** then checks the time since the last update. If the new contact observation time is later than the track head update time, the contact is used to update the track head using a set sequence of object calls. This sequence contains the Kalman filter procedures as well as two routines which maintain the latitude and longitude relationships on a spherical earth and proceeds as follows:

- **ioumotion:** Performs the IOU prediction step and updates the N/S and E/W offset distances in the track head state to the time of observation of the contact. These offsets effectively perform the prediction in a plane tangent to the earth's surface at the reference latitude and longitude point of the track head.

- **change-tangent:** Finds the relationship of the iou predicted position and the contact point in a plane tangent to the earth at the contact's reference point vice the track head's reference point used to compute the prediction in **ioumotion**. These matrices are stored as **state-cov-old**. Since the covariance is circular, there is no need for the sequence which rotates it.

- **make-state:** Takes the contact data and produces z_t which is called simply **mean** and its covariance R which is called **var**. These two matrices are stored in **state-cov-ctc**.

- **filter:** Takes **state-cov-old** and **state-cov-ctc** and uses Equations 2.14 and 2.15 to compute x_{k+1} and Σ_{k+1}. The result is the Kalman filter update performed in a plane tangent to the earth at the contact's reference point.

- **center-tangent:** The latitude and longitude of the position produced in **filter** is projected back onto the earth from the tangent plane.

- **update-track-head:** The new updated state and covariance are made the new track head. The contact used to perform the update provides the track-head contact data.

- **update-track-tail:** Sorts the contacts making up the track in time order, oldest to most recent (1 to n). If there is only a single contact, it is designated the track tail. If there are five or fewer contacts and the oldest contact is already the tail it remains so; otherwise the oldest contact is made the new tail. If there are more than five contacts, it takes number contacts, subtracts five and the contact whose number equals the result is used to update the track tail using the procedure detailed above. The contact used to update the tail provides the track tail contact data. All contacts older than the tail are then pruned from the track.

Considerable computation time can be saved by simulating a portion of the above process, rather than actually carrying out all the steps. A revised update procedure based on using the updated variance (Σ_{k+1}) as an approximation to the distribution of updated states (x_{k+1}) is proposed. Rather than produce the contact position latitude and longitude randomly using the sensor variance as a distribution and processing it using the sequence above, the latitude and longitude are chosen by computing the updated variance, which can be done prior to scenario execution. When it is needed, the distribution is centered on the target's true position, and used to generate the updated states directly by random draw. This new procedure analyzed in detail in Chapter IV.

Time delays incurred at communications nodes enroute to the fusion center may cause the observation time to fall between the track head update time and the track tail update time. In that case a modified procedure is followed .

- If the time of observation for the contact is later than the tail update time or if the track does not yet consist of five contacts, then the contact is appended to the track and the contacts are sorted by time of observation. If the contact is older than the tail and the track already consists of five contacts, it is ignored and the procedure stops.

- If the observation time is later than or equal to the tail update time, the tail is used as the oldest contact, if the new contact is older, then **compute-mean-var** is used to make the new contact the oldest.

- The oldest contact is then updated using the procedure described above using the next contact in the track, in time sequence, as the "contact". The result of this becomes the "old contact" and the procedure is repeated, updating each contact in the track in sequence until the head is updated. The resulting "forward filtered" track closely approximates the track head which would have resulted if the time-late contact had been received in sequence.

The calculations performed in **ioumotion** and **filter**, all assume that elliptical AOU's are being produced by the Detection Reporter and General Sensor objects. The object **make-state** can be simplified in the manner as **compute-mean-cov** as it performs the same operations. The reduced forms of the equations in those objects are computed below.

A. REDUCED COMPLEXITY IOU PREDICTION

The matrix equations for calculating the IOU predicted state and covariance are given by Equations 2.9 and 2.10. Using the symbolic processing capability of the MathCad 3.0 (copyright 1991 MathSoft, Cambridge MA) computer software package,

the matrix operations were evaluated. The large number of zero terms and high symmetry of the ϕ and Q matrices resulted in simple expressions for the elements of the state and variance predictions. Portions of the **calc-mot-mat** object which compute f(0,0), f(0,2), f(2,2) for positive time as well as $\phi(0,2)$ and $\phi(2,2)$ can be used to produce the terms of the following equations used to compute **new-var:**

$$\hat{x}_{k+1}=newmean=\begin{bmatrix} E/W\ pos_{k+1} \\ N/S\ pos_{k+1} \\ E/W\ vel_{k+1} \\ N/S\ vel_{k+1} \end{bmatrix}=\begin{bmatrix} \hat{y}_{k+1} \\ \hat{x}_{k+1} \\ \hat{Vy}_{k+1} \\ \hat{Vy}_{k+1} \end{bmatrix}=\begin{bmatrix} y_k+phi(0,2)\ Vy_k \\ x_k+phi(0,2)\ Vx_k \\ Vy_k\,phi(2,2) \\ Vx_k\,phi(2,2) \end{bmatrix}, \tag{3.2}$$

where:

$$\phi(0,2) = \frac{1}{\beta}(1-e^{-\beta t}) \quad and \quad \phi(2,2) = e^{-\beta t}; \tag{3.3}$$

and:

$$\hat{\Sigma}_{k+1}=newvar=\begin{bmatrix} \hat{Va}_{k+1} & 0 & \hat{Vb}_{k+1} & 0 \\ 0 & \hat{Va}_{k+1} & 0 & \hat{Vb}_{k+1} \\ \hat{Vb}_{k+1} & 0 & \hat{Vc}_{k+1} & 0 \\ 0 & \hat{Vb}_{k+1} & 0 & \hat{Vc}_{k+1} \end{bmatrix}, \tag{3.4}$$

where:

$$\begin{bmatrix} \hat{Va}_{k+1} \\ \hat{Vb}_{k+1} \\ \hat{Vc}_{k+1} \end{bmatrix} = \begin{bmatrix} Va_k+2\,(phi(0,2))\ Vb_k+phi(0,2)^2\ Vc_k+f(0,0) \\ (Vb_k+phi(0,2)\ Vc_k)\ phi(2,2)+f(0,2) \\ phi(2,2)^2\ Vc_k+f(2,2) \end{bmatrix}, \tag{3.5}$$

and with:

$$
\begin{aligned}
f(0,0) &= \frac{\sigma^2}{\beta^2}\left[t-\frac{2}{\beta}(1-e^{-\beta t})-\frac{1}{2}(1-e^{-2\beta t})\right], \\
f(0,2) &= \frac{\sigma^2}{\beta^2}\left[(\frac{1}{2}-e^{-\beta t})+(\frac{1}{2}e^{-2\beta t})\right], \\
f(2,2) &= \frac{\sigma^2}{2\beta}(1-e^{-2\beta t}).
\end{aligned}
\tag{3.6}
$$

The f terms given here are the ASSET versions of (2.36), (2.37) and (2.38).

B. REDUCED COMPLEXITY KALMAN UPDATE

In a manner analogous to that used in the preceding treatment of the IOU update, the matrix equations for the Kalman update were evaluated. The result is:

$$
Kalman\ Gain = \begin{bmatrix} Ka_{k+1} & 0 & Kb1_{k+1} & 0 \\ 0 & Ka_{k+1} & 0 & Kb1_{k+1} \\ Kb2_{k+1} & 0 & Kc_{k+1} & 0 \\ 0 & Kb2_{k+1} & 0 & Kc_{k+1} \end{bmatrix}. \tag{3.7}
$$

where:

$$
\begin{aligned}
Ka_{k+1} &= \frac{1}{D}\left[\hat{Va}_{k+1}(\hat{Vc}_{k+1}+\frac{\sigma^2}{2\beta})-\hat{Vb}_{k+1}{}^2\right], \\
Kb1_{k+1} &= \frac{1}{D}\left[\hat{Vb}_{k+1}\frac{1}{4}r^2\right], \\
Kb2_{k+1} &= \frac{1}{D}\left[\hat{Vb}_{k+1}\frac{\sigma^2}{2\beta}\right], \\
Kc_{k+1} &= \frac{1}{D}\left[\hat{Vc}_{k+1}(\hat{Va}_{k+1}+\frac{1}{4}r^2)-\hat{Vb}_{k+1}{}^2\right], \\
D &= (\hat{Va}_{k+1}+\frac{1}{4}r^2)(\hat{Vc}_{k+1}+\frac{\sigma^2}{2\beta})-\hat{Vb}_{k+1}{}^2.
\end{aligned}
\tag{3.8}
$$

Using these gain values the corresponding updated variance terms are:

$$
\Sigma_{K+1} = \begin{bmatrix}
Va_{k+1} = Ka_{k+1}\frac{1}{4}r^2 \\
Vb1_{k+1} = Kb1_{k+1}\frac{\sigma^2}{2\beta} \\
Vb2_{k+1} = Kb2_{k+1}\frac{1}{4}r^2 \\
Vc_{k+1} = Kc_{k+1}\frac{\sigma^2}{2\beta}
\end{bmatrix}. \tag{3.9}
$$

The updated state is written in terms of residuals which are the differences between the contact states and predicted states:

$$Residuals = \begin{bmatrix} Ry \\ Rx \\ RVy \\ RVx \end{bmatrix} = \begin{bmatrix} zy - \hat{x}y \\ zx - \hat{x}x \\ zVy - \hat{x}Vy \\ zVx - \hat{x}Vx \end{bmatrix}. \qquad (3.10)$$

$$\begin{bmatrix} y \\ x \\ Vy \\ Vx \end{bmatrix} = \begin{bmatrix} \hat{y} + Ka\,Ry + Kb1\,RVy \\ \hat{x} + Ka\,Rx + Kb1\,RVx \\ \hat{Vy} + Kb2\,Ry + Kc\,RVy \\ \hat{Vx} + Kb2\,Rx + Kc\,RVx \end{bmatrix} \qquad (3.11)$$

The reduced form equations for the state are included here, but are unnecessary if the estimation of the state is to be used. The accuracy of the estimation technique is analyzed in the following chapter.

IV. MODIFICATIONS TO THE ASSET TRACKER

The tracker currently used in ASSET makes two major assumptions relating to detection modelling:

- Sensor AOU's are circles vice ellipses,
- The velocity variance inherent in the sensor is ignored when making the filtered variance calculation for position and velocity contacts.

Figure 2 shows the geometry of a typical filter operation:

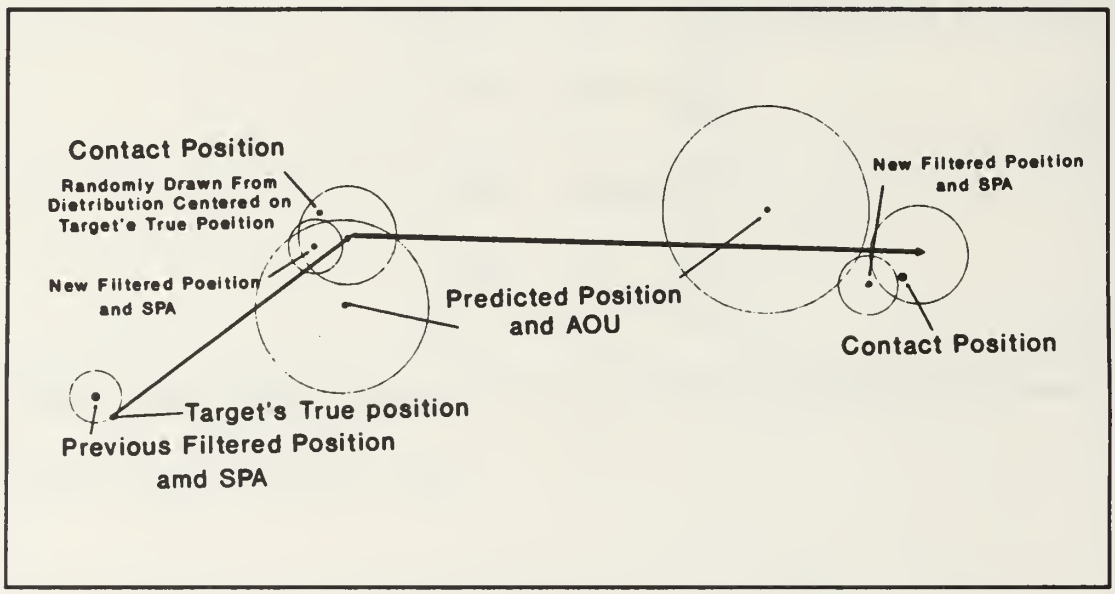


Figure 2: Geometry of Filter Update Operation.

The equations which exactly determine the position of the center of the SPA are given in (3.12). Given that the simulation knows the true target location and making several assumptions detailed below, it is proposed to bypass the Kalman state update

step and estimate one possible position of the SPA based on the relationship between the SPA center and the true target position. If the true position is distributed somewhere in the SPA, then from the point of view of the target, the SPA center is likewise distributed about its position. This allows an estimate of a SPA center to be generated by drawing a random variable from a distribution with standard deviation equal to the square root of the filtered variance in position with a mean centered on the target's true position. This uses the SPA "in reverse" to find a possible position for its own center based on the targets true position. Provided the necessary assumptions are met, this will provide a reasonable approximation to the filtered track over a series of observations. It also allows the variance values to be precomputed and drawn from a table vice calculated when needed. These three assumptions are:

- The mean predicted position and velocity must be very close to the target's true position and velocity so the AOU of the predicted position is concentric with the contact AOU, or nearly so. This assumption relates to the motion model in the tracker simulated.
- It is sufficient to model the behavior of detections in the mean, rather than explicitly computing them individually. This assumption relates to the interaction between the correlator and the tracker.
- The distribution of filtered states has its mean at the target's true position and a variance equal to the steady-state value of *Va* (3.10).

The first assumption sets a requirement for the accuracy of the motion model. If the predicted position is consistently offset a considerable distance from the target's true position and the Kalman gain is small, then the center of the filtered position

distribution will be drawn off the target's true position. This can be shown by rearranging the terms in Equation (3.12) to yield for a single dimension:

$$y = \hat{y}(1-Ka) + Ka\,Zy + Kb1\,(ZVy - \hat{Vy}) \qquad (4.1)$$

The filtered position y approaches the contact position Zy as Ka goes to one and moves off to the predicted position $\hat{y}$ as Ka goes to zero. This case must be avoided to prevent significant inconsistencies between the distribution of actual filtered positions and the estimates generated. Working to keep this relationship is the effect shown in (2.5) which acts to push the position toward the observed position as the time interval increases for a given observation AOU size and the IOU prediction variance gets large compared to it.

This effect is not easily identifiable in the Kalman gain equations, but increasing the time interval increases the size of the predicted AOU and increases Ka for a given observation AOU size. This counteracts the tendency of the predicted position to recede to the previous filtered position as the time interval increases and keeps the required relationship intact for cases where the time interval between observations is large. The maximum error will occur when the two AOU's are nearly the same size and the value of Ka is approximately .5-.65. This assumption is also involved in the computation of the filtered velocity, where it is assumed that the difference between the predicted and observed position multiplied by gain $Kb2$ is small.

The second assumption requires acceptance of a lack of consistency between the correlator and the tracker models. The ACT-ASSOC module uses the Kalman filter

for calculation of the spatial measure of correlation it uses to associate contacts to tracks. The effect of using equally representative, but different, implied contact locations by the correlator and tracker to do contact association and track updating has not been thoroughly examined. The fact that the filtered position produced by this method does not relate to the contact position used for track association does not appear to be a serious problem, given the Monte Carlo nature of the simulation. The draw used by the correlator can be looked at as determining whether the proper track assignment is made, a false assignment made or no assignment made. If the contact is drawn from the proper distribution, the probability that any single draw results in any of these occurrences is the same whether or not the tracker goes on to use the contact to update the track chosen.

The third assumption is again linked to the accuracy of the motion model. The result of making this assumption is shown in Figure 3. The first part is essentially the effect of assumption one being true. The second part is required because the variance in distribution of the actual filtered position is affected by the speed of the actual target. For target speeds less than the modelled speed, the actual distribution is tighter than the estimated value. The opposite is true if the target speed is faster than the model. The implication of this assumption is that the target is exactly performing the motion modelled and the tracker does not respond as the actual Kalman filter does to deviations from the expected motion. Since the range of velocities the submarine targets may have is small, this deviation is typically less than fifteen percent of the SPA size.

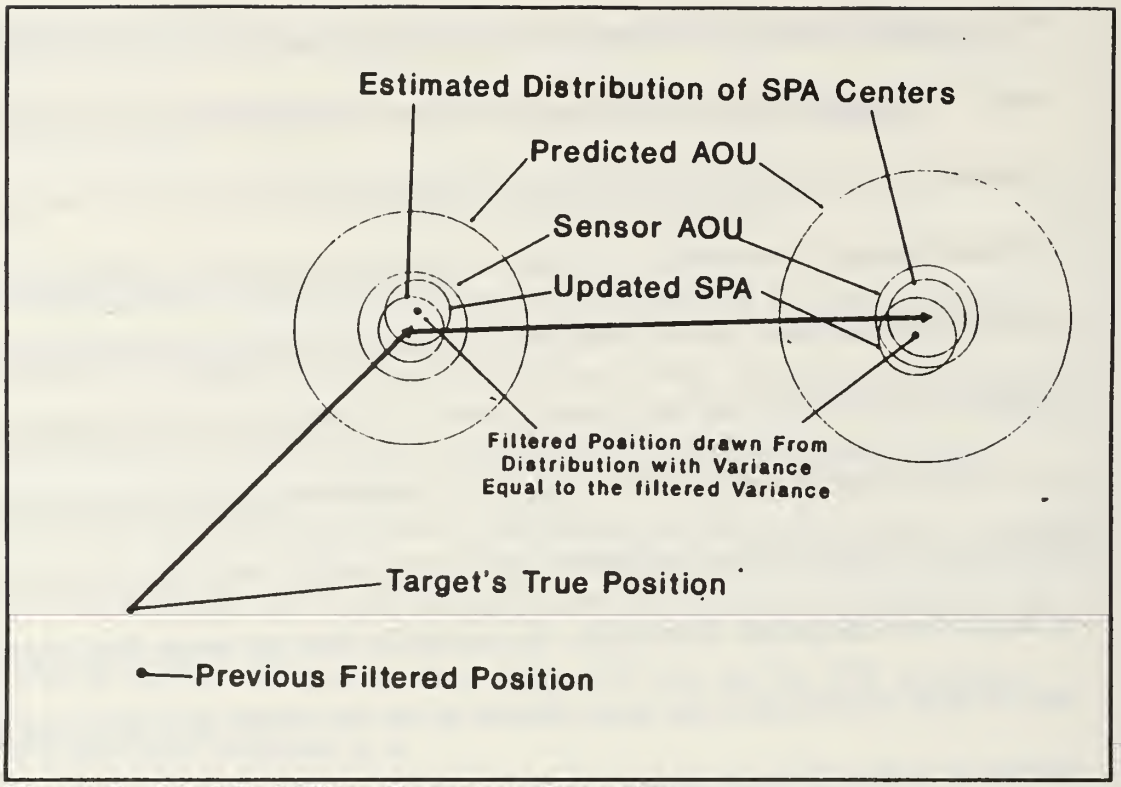


Figure 3: Modified Filter Operation Geometry.

A. PROPOSED MODIFICATIONS

Implementation of these changes in the code is a part of a separate effort to port ASSET to the Sun UNIX workstation. A listing of the applicable code is contained in Appendix B. The **update-track** object described in Chapter III ia the primary object to be modified to implement these changes. Its function can be split between those used to precalculate the variance values and those used to estimate the velocity if precalculation is desired. This involves computing the three steady state variance values for each sensor AOU size, for representative values of the contact interarrival

time. Other objects to be changed are mentioned below and included in the Appendix B as well. A summary of these modifications include:

- **ioumotion:** The three element values of the f matrix (f(0,0), f(0,2) and f(2,2)) and the two ϕ matrix values ($\phi(2,2)$ and $\phi(0,2)$) are computed as before, but not set in matrix form. The values are used to compute the three element values of the predicted variance using Equation (3.6). The three computed values will be used to compute the steady-state filtered variance values which are stored in a table for future reference. A table of the number of iterations needed to reach steady state versus interarrival time is detailed in Appendix C.

- **change-tangent:** This object is not required.

- **make-state:** The random numbers used to determine the contact's latitude and longitude (from the **contactPositionDraw** function called by the Detection Reporter) can be retained, but as discussed above this seems unnecessary. The two values of the *R* matrix are all that need be calculated by this object.

- **filter:** The values computed in **ioumotion** are used to calculate the three elements of the filtered variance using Equation 3.10 for representative values of the contact report interarrival time. A graph of the steady state gain *Ka* versus interarrival time is contained in Appendix D. The Kalman gains are computed in order to find those values and will be used below to set the range of operation of the filter.

This set of routines comprise the pre-processing module and can be separated from the rest of MTST so it computes the necessary data at the start of the simulation. This requires computing the three variance values for 33 interarrival times, giving approximately .03 between succesive table values from .03 to .97. This requires 100 values be tabulated for each AOU defined. Retrieving values from a data array of perhaps 1500 elements appears to be significantly faster than calculating the values individually each time the filter is called.

The following routines perform the necessary computations at the time the filter operation is performed:

- **center-tangent:** This object is modified to call the function **contactPositionDraw** and, using the filtered variance drawn from the table computed above, as positional variance choose a center for the SPA. The distances involved to not require the use of the spherical earth routine in **contactPositionDraw**. The new latitude and longitude can be calculated using 1° of latitude=60.1077nm and 1° of longitude=60.1077*cos(latitude)nm. The error in this estimate is less than one percent out to ranges of 600nm at non-polar latitudes. The filtered velocity can be estimated by the following equation:

$$\begin{aligned} Vy_{t+1} &= Vy_t e^{-\alpha t}(1-K3)+K3\,ZVy_t \\ Vx_{t+1} &= Vx_t e^{-\alpha t}(1-K3)+K3\,ZVx_t \end{aligned} \qquad (4.2)$$

- **update-track-head:** The position and velocity computed in **center tangent** and the new variance values are used as before to update the track-head. The state consists of the position latitude and longitude and the corresponding velocity components, while the covariance consists of the three steady-state values of the filtered variance.

- **update-track-tail:** This is modified in the same way as **update-track-head** to account for the new form of the state and covariance.

The result is a close approximation to the variance values obtained from the full implementation and a position that represents the result of some combination of predicted position and contact position that was not specifically identified and operated on. The distribution of estimated filtered states closely approximates that of actually computed states. The specifics of this comparison are detailed below.

B. ACCURACY OF THE MODIFIED TRACKER

The assumptions underlying this new procedure introduce deviations from the full implementations. As discussed above, the error in the positioning of the SPA about the target's true position error is maximum for *Ka* approximately .5-.65. The mean value of this error, assuming the worst case of a non-maneuvering target, can be calculated using Equation (3.3) to find distance from the old filtered position to the new observation position. Since the observations are centered on the true target position, the mean of this value can be calculated using the true target position. The results are shown in Appendix E for favorable and unfavorable conditions. From these results it can be seen that the error is excessive for targets at high speed if the time between detections is very long. This quickly drops off however, and for more reasonable time intervals of less than one hour, the relative error, even for high speed targets, is negligible.

Assumption three must also be valid for the approximation to be reasonable. The validity of the first part of the assumption is shown above. This requires that the variance of the distribution of filtered positions be equal to the filtered variance. Appendix F presents a comparison of filtered position distributions and filtered AOU sizes for various target speeds, computed using the model in the Appendix G. The data in Appendix F is the result of modelling 48 hours worth of tracking data or no more than 300 observations. The simulated target conducted a random tour consisting of ten course changes executed at time intervals with an exponential distribution and a

mean of 4 hours. The target's speed was either 3 knots, 10 knots or 30 knots. The standard ASSET IOU parameters $\beta=.25$ and $\sigma=10\beta^{1/2}$ were used. The results of 30 different iterations were averaged to determine the mean values.

The results indicate that for long time intervals between contacts, and speeds near the modeled one, the results are fairly good. Excessive deviations are exhibited when the target's speed is far in excess of the modeled speed. Even in the case where the actual target speed was ten knots, the damping of the velocity is apparent in the filtered mean speed. When this is combined with the offset error described above, errors of up to sixteen percent can result. While this is a large error, the average value this error takes through the range of common time intervals between observations (.1 hours to 1 hour) and possible contact speeds, is closer to five percent.

One source of this error can be seen by comparing (3.10) and (3.12). While the single Kalman gain *Ka* is used to compute the value of the variance, the actual distribution is computed using both *Ka* and *Kb1*. For the range of AOU sizes associated with submarines and MPA assets (at worst 25nm) leaving the SPA unadjusted amounts to an error in standard deviation of the estimated distribution of less than 1nm. The error is typically conservative, increasing the SPA size by the error, and is largest when the time interval between contacts is small. This produces errors which, while a significant percentage of the AOU size, are actually quite small. A notable exception is the decrease in the standard deviation in situations where the

target speed is high, counteracting some of the IOU process' tendency to underestimate the targets speed in this case.

The failure of the estimate to account for the target's velocity causes the resultant track to be erratic. This effect worsens as the distance traveled between observations gets small with respect to the AOU size. The actual filter exhibits consistant errors in this case, while the estimate zig-zags wildly around the true target track. Since the linearity of the track is not critical to any part of ASSET except possibly the correlator, the effect is minimal. For any single point the estimate produces, a track could exist which includes it. The random draw combines elements from all these hypothetical tracks into a single collection which has the proper pointwise relationship to the real track. Thus, while specific combinations of observation AOU size and observation interval result in poor performance, others result in improved performance.

The performance of the modified filter is reasonably consistent with the original filter operation. A 3-5 nm error in a 25 nm AOU does not significantly impact the cueing or search efforts of prosecuting platforms and tracker performance is actually improved for cases where the target is transiting, moving at high speeds without maneuvering. The errors present in the modified tracker appear to have no negative impact on the operation of the rest of the simulation and while introducing moderate errors in tracking maneuvering targets, balances that with improved performance against non-maneuvering ones. Appendix G contains examples of the original filter's

tracking ability and that of the estimate for a variety of sensor AOU sizes and mean interarrival times.

C. LIMITING THE RANGE OF FILTER OPERATION

An interesting effect apparent from Appendix F, is the limiting nature on the filtered position distribution of the contact AOU on the one hand and the predicted AOU on the other. This situation stems from the same effect that moves the filtered position from the observation position toward the predicted position. This can be seen directly by looking at the values of the Kalman gains as a function of range for a given time interval between observations. Examples of these relationships are shown in Appendix H. As can be seen from these graphs, if the observation AOU radius in nautical miles is limited to being less than 150 times the time interval in hours, the *Ka* value remains above the "flat" portion of the curves, approaching zero. Observations which arrive at the tracker such that the interval since the last contact is too small compared to the AOU size will be filtered using primarily the prediction information. The effect of the contact is largely ignored. This is the filter's way of saying that the information value of the contact is not high enough to be considered of value in computing the filtered position.

Such contacts can be ignored as there is little difference between the predicted position and variance and the resultant state and variance. A lower limit of 1nm on the observation is hardwired into the **make-state** and **compute-mean-cov** objects. If the time between contacts is too short, little is gained unless the AOU is tiny. The

1nm limit implies that the minimum time interval between contacts is around .01 hours to allow the contact data to be assimilated into the track. If the contact makes no contribution to the track update, there is no reason to perform the filter operation. The effect of this can be seen in the data in Appendix F, where the predicted and filtered values are very close for small time intervals and large observation AOUs. Thus in dense contact environments, with contacts arriving at the fusion center at intervals of less than 30 minutes, any contact based on an AOU larger than 150 times the time since the last contact in hours, can be ignored and not filtered. This value of 150 corresponds to a Kalman Gain limit of approximately .1. The exact value is rather arbitrary so long as it not too far into the portion of the curves that change quickly with variation in observed AOU size. This could be made a user input to limit the resolution of the tracker and speed execution time during scenario development and testing, and then reduced to provide more accurate track information when the data is actually collected.

In similar fashion tracks which have not been updated in a long time, do not contain enough information to affect contacts that arrive with AOUs less than 5-10 times the time interval since the last observation. If the value of *Ka* is very close to the "flat" portion approaching one, the predicted value is ignored in favor of the contact data. This allows similar saving of filtering operations in sparse track conditions with accurate sensors. If the time interval between contacts is expected to be greater than one hour and the contact AOUs are small, using the raw contact to

update the track is a reasonable approximation if the observation AOU size is smaller than five times the time between contacts in hours.

When these two "gating" techniques are utilized, substantial numbers of filtering calls can be eliminated. This can be particularly true in scenarios covering large ocean areas with many possible targets. During the initial stages of the campaign, as search assets slowly get cues for very quiet enemy tactical platforms, detections will not be made until the enemy is at close range. This will result in observations with relatively small AOUs coming in at long time intervals. As cues are dispatched and friendly forces converge on the enemy, even a modest number of platforms soon begin producing contact streams that arrive at rapidly decreasing intervals. Putting limits on when the filter is invoked will prevent the simulation from wasting efforts computing a SPA that does not really use much of the contact information.

V. CONCLUSION

From a discussion of the construction and use of the Kalman filter based Maneuvering Target Statistical Tracker it has been shown that the tracker implementation in ASSET is not currently matched well to the types of inputs it receives. This results in the calculation of many duplicate values, the performance of unnecessary matrix manipulations and the calculation of values in situations when little useful information is gained. Improvements to the tracker aimed at decreasing the computation time which were discussed include:

- Use of the equations detailed in (3.3) through (3.12) to compute the actual filtered variance and position.
- Filter the variance of the contact distribution about the target prior to conducting a random draw, thus producing a position drawn from a distribution approximating that of the actual filtered positions variance. Since the observations play no part in calculating this variance, appropriate values for the sensor AOUs can be precomputed for a range of expected time intervals.
- Limit the situations where the filter is actually used to those which will result in a meaningful amount of information being extracted.
- Using planar estimations of the latitude and longitude computations rather than perform the conversions on a "spherical earth".

When taken together these modifications greatly reduce the complexity of the calculations required to update a track after an observation. They can also be put

under user control so the level of fidelity can be adjusted based on the requirements of the architecture and the stage in development of the architecture the analyst is at.

Areas for future work in this area include:

- Formulating a faster way of estimating the steady state gains for use in a table of pre-calculated filtered variances.
- Analysis of the filtered position distributions of other types of trackers to determine if a better method of approximation can be determined, one which better accounts for target velocity.
- Formulating a simple, recursive estimate of the Kalman gains for on-the-fly calculation. An attempt at this by using limiting values as the time interval got small yielded very good steady state results, but divergence and chaotic behavior for small AOU sizes (< 10nm radius) precluded its use iteratively.
- An analysis of the algorithms in the correlator to determine if all of its algorithms are necessary, or if a simple probability of correct/false/no correlation could be substituted. If appropriate, determine what those probabilities should be.
- Modelling how the AOU size should change with respect to range and bearing from the searcher. A strict linear bearing error vs range is a simple way of solving the problem of constant sensor AOU size. The real situation is not quite so simple. This requires computation of the beam sizes and beam distribution of a towed array and the effect of TMA techniques on determining the position course and speed of the target. This would allow for proper calculation and filtering of the sensor's variance in velocity measurement as well.

The last example opens up a plethora of possibilities for analysis of the parameters required for input to ASSET. A compendium of data on everything from communication network modelling, to aircraft maintenance data analysis, to platform on platform engagement simulation, and so on. Well researched values for the present and expected future interactions of platforms and their supporting

infrastructures need to be produced if ASSET and simulations like it are to be available to the average user.

ASSET has the potential to be a very effective analysis tool. A great deal of work must be expended to provide the necessary data to allow its practical use. In its present form it requires so much expert knowledge on so diverse a collection of topics, it is doubtful one person could perform meaningful analysis without an inordinate amount of time spent in research. Continued evolution in the functionality of the simulation will provide increased applicability and accuracy of results. Increased streamlining of the computational algorithms will be necessary as this occurs to keep it running on a desktop workstation in a reasonable period of time. Striving for improvement in performance however, without providing a ready source of data will most probably preclude widespread use of the simulation.

APPENDIX A: CHARACTERIZATION AND ESTIMATION OF ϕ AND Q.

The following equations represent the elemets in the transition matrix, phi, and the IOU process noise, Q. Approximation equations which can be used to more quicly calculate the values are also given.

$$\phi = \begin{bmatrix} 1 & 0 & \phi 2 & 0 \\ 0 & 1 & 0 & \phi 2 \\ 0 & 0 & \phi 3 & 0 \\ 0 & 0 & 0 & \phi 3 \end{bmatrix} \quad Q = \begin{bmatrix} q1 & 0 & q2 & 0 \\ 0 & q1 & 0 & q2 \\ q2 & 0 & q3 & 0 \\ 0 & q2 & 0 & q3 \end{bmatrix}$$

$t = 1, 10 .. 5000 \quad time_t = (t) \cdot .01 \quad time_0 = .1$

$n = 1, 10 .. 300$

$V = 10 \quad \alpha = .25 \quad \sigma = V \cdot \sqrt{\alpha}$

Actual value:

$$\phi 2_t = \frac{1}{\alpha} \cdot \left[1 - \exp\left[- \alpha \cdot time_t \right] \right]$$

$$\phi 3_t = \exp\left[- \alpha \cdot time_t \right]$$

$$q1_t = \frac{\sigma^2}{\alpha^2} \cdot \left[time_t - \left[\frac{1}{\alpha} \cdot \left[\left[2 \cdot \left[1 - \exp\left[- \alpha \cdot time_t \right] \right] \right] - .5 \cdot \left[1 - \exp\left[- \alpha \cdot 2 \cdot time_t \right] \right] \right] \right] \right]$$

$$q2_t = \frac{\sigma^2}{\alpha^2} \cdot \left[.5 - \exp\left[- \alpha \cdot time_t \right] + \left[.5 \cdot \exp\left[- \alpha \cdot 2 \cdot time_t \right] \right] \right]$$

$$q3_t = .5 \cdot \frac{\sigma^2}{\alpha} \cdot \left[1 \quad \exp\left[\alpha \cdot 2 \cdot time_t \right] \right]$$

Estimated Value:

$$\phi 2\varepsilon_t = \frac{-1}{2} \cdot time_t \cdot \left[-2 + \alpha \cdot time_t \right]$$

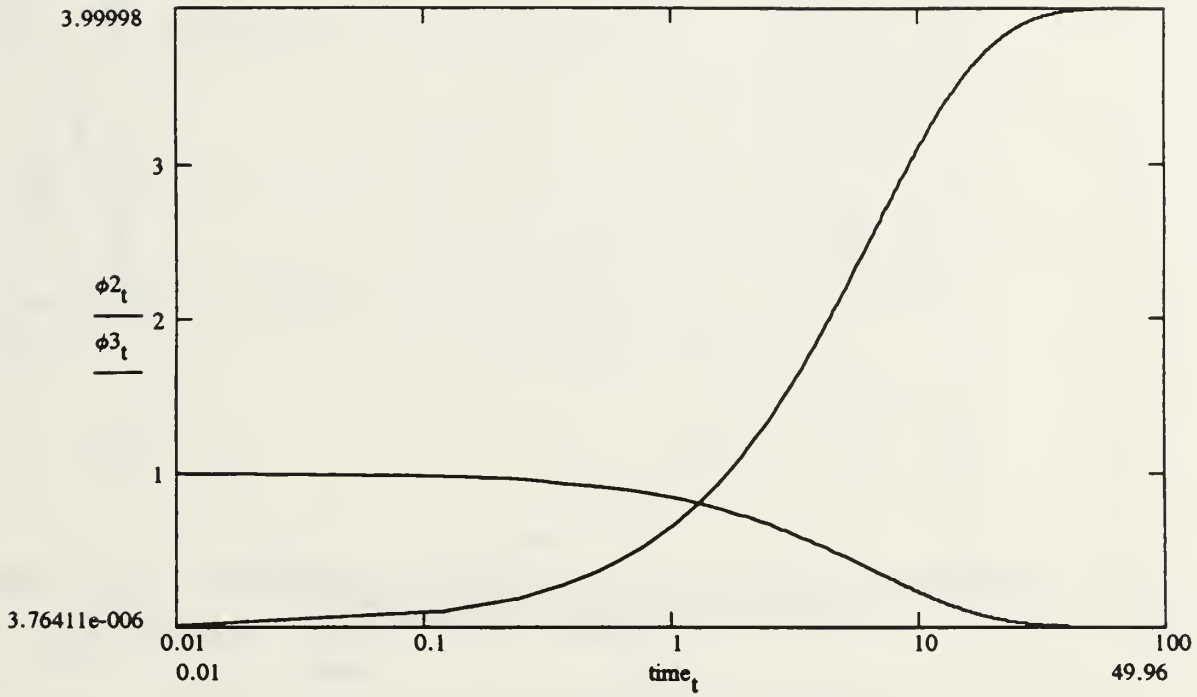
$$\phi 3\varepsilon_t = 1 + \frac{1}{2} \cdot \alpha \cdot time_t \cdot \left[\alpha \cdot time_t - 2 \right]$$

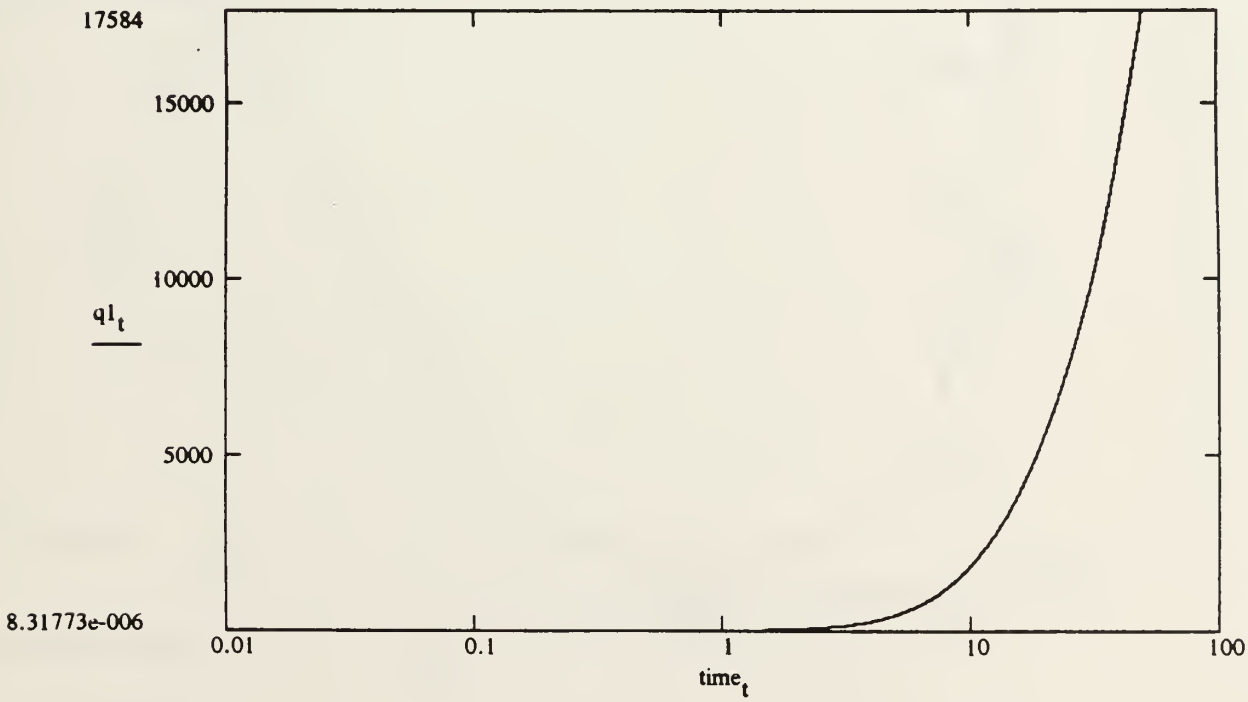
$$q1e_t = \sigma^2 \cdot \left[time_t \right]^3 \cdot \left[\frac{1}{3} - \frac{1}{4} \cdot \alpha \cdot time_t \right]$$

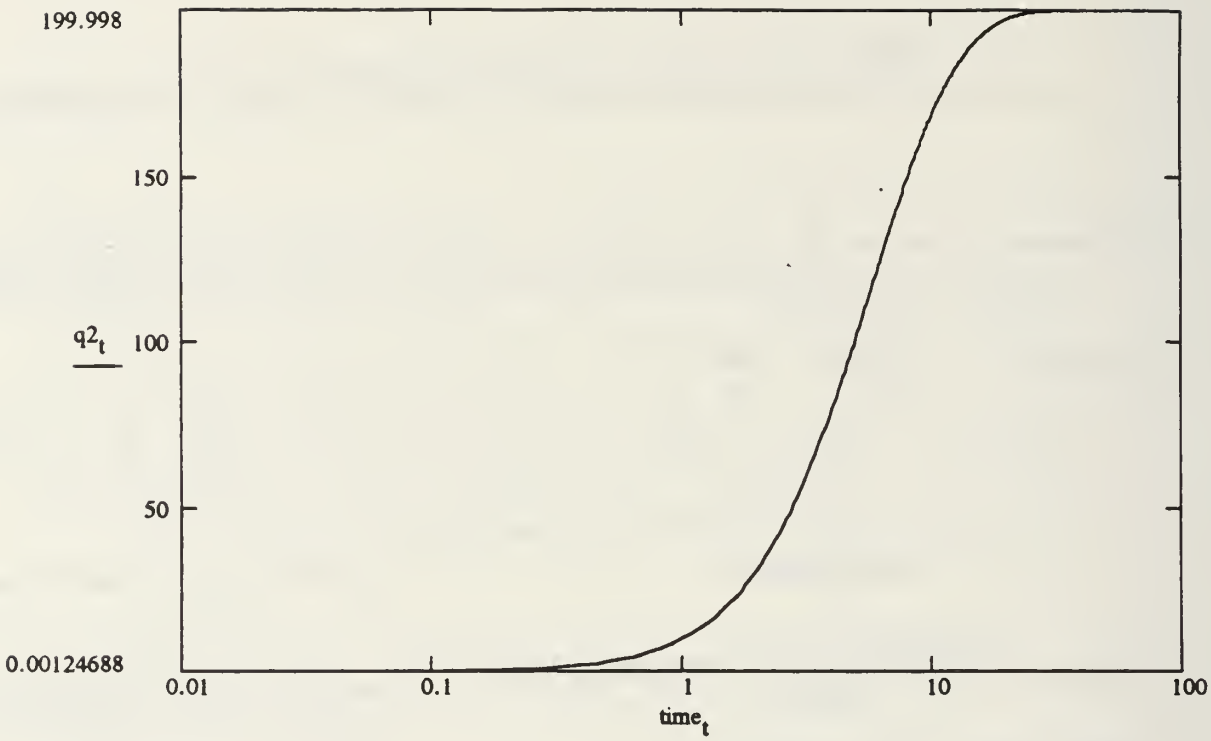
$$q2e_t = \sigma^2 \cdot \frac{1}{2} \cdot \left[time_t \right]^2 \cdot \left[1 - \alpha \cdot time_t \right]$$

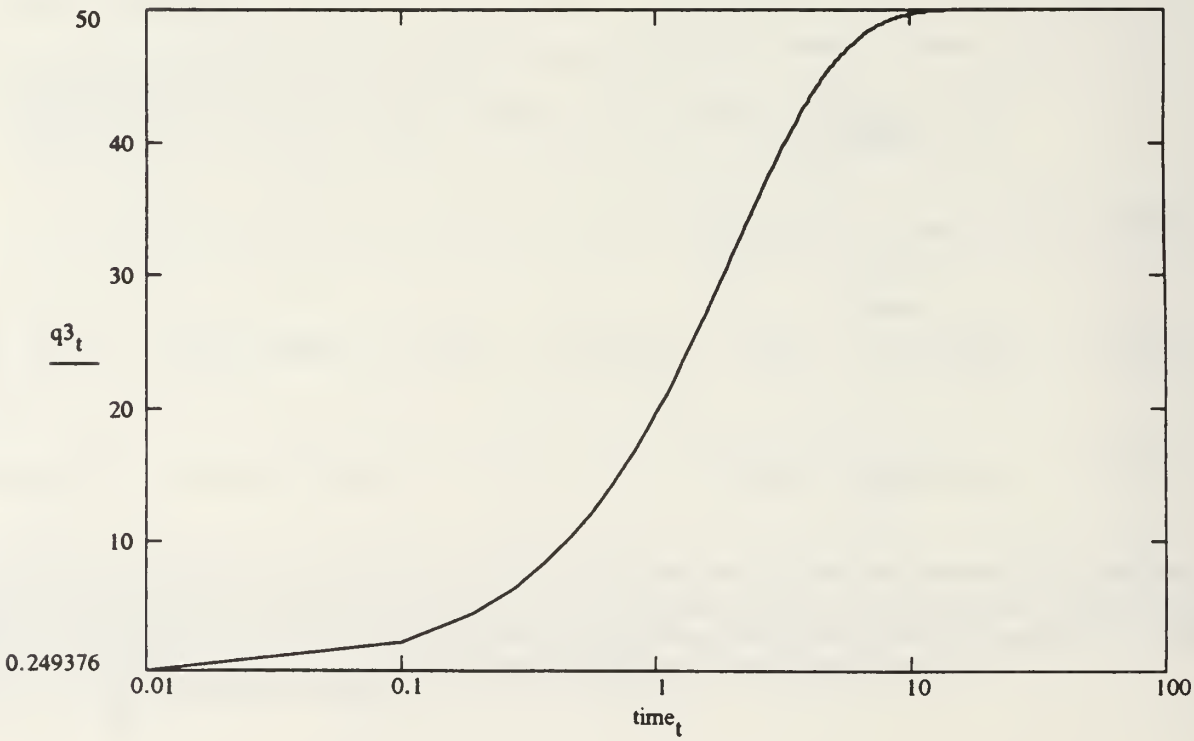
$$q3e_t = -1 \cdot \sigma^2 \cdot time_t \cdot \left[1 + \alpha \cdot time_t \right.$$

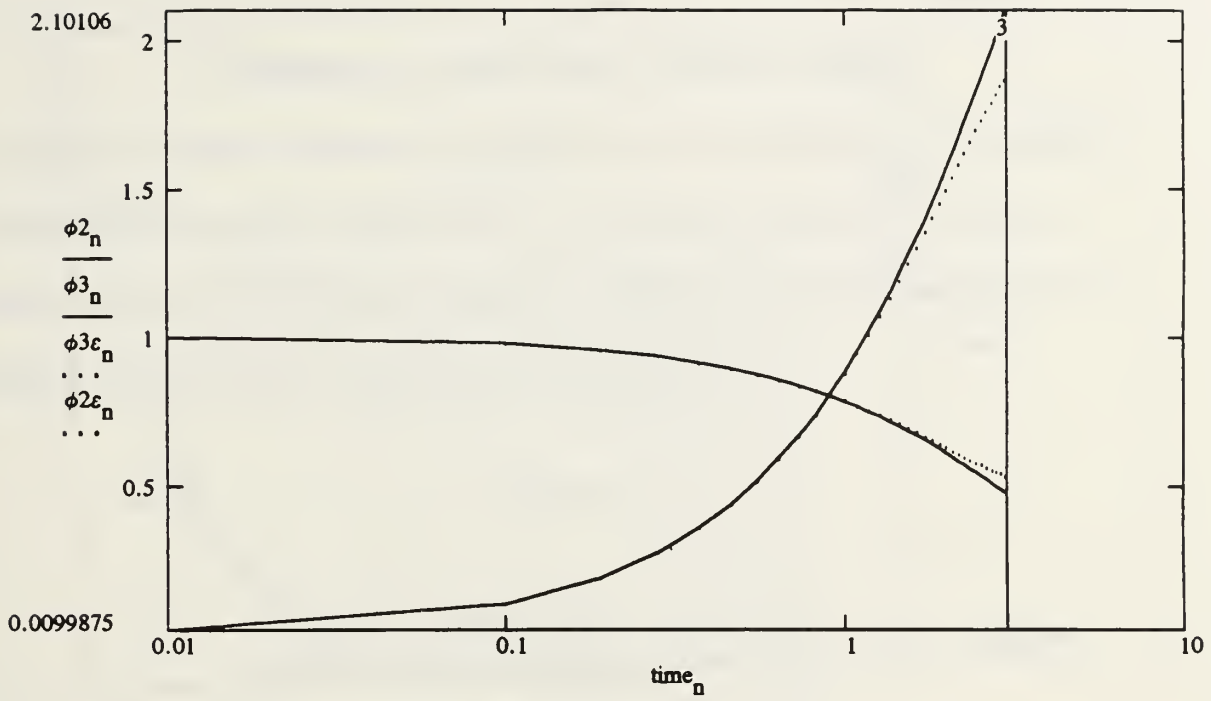
The graphs which follow show the behavior of these functions over various time intervals. The variable "time" is equal to the elapsed time from the last contact's time of observation to that of the present contact, in hours. The estimated values are computed using the first two significant terms in the series expansion of the exponential terms. These estimates are very accurate for interarrival times of two hours or less.

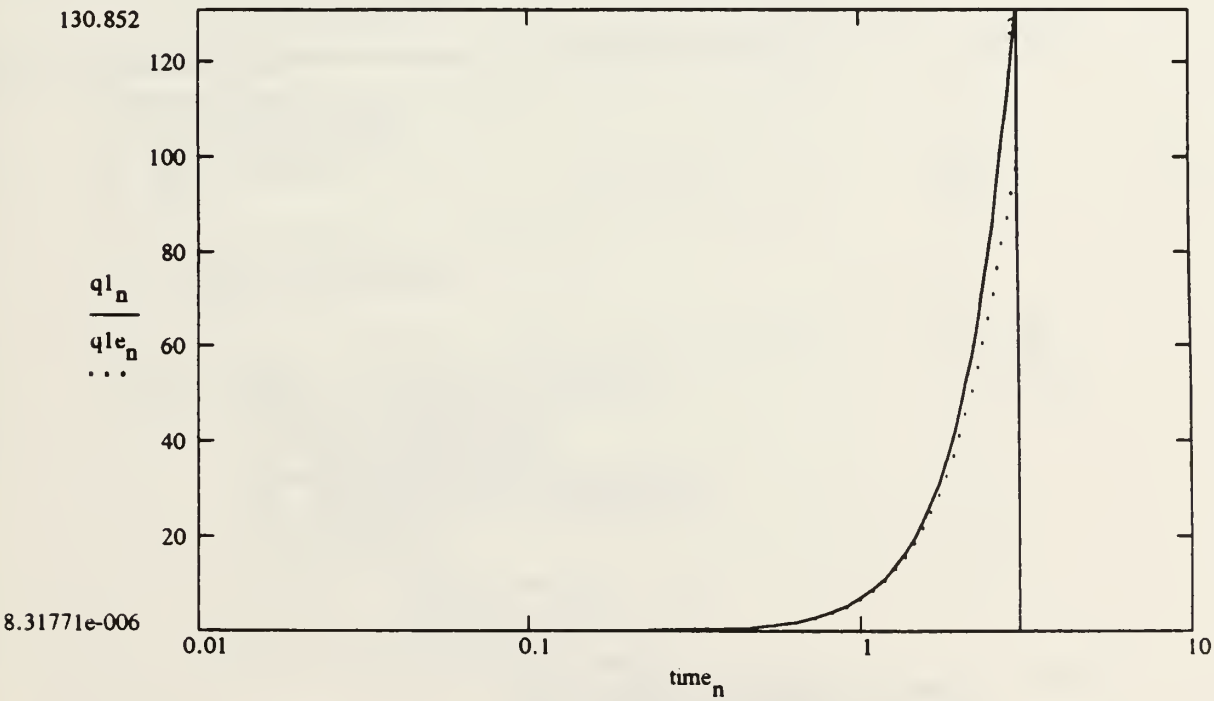

3.99998
3
$\frac{\phi 2_t}{\phi 3_t}$
2
1
3.76411e-006
0.01
0.1
1
10
100
0.01
time$_t$
49.96

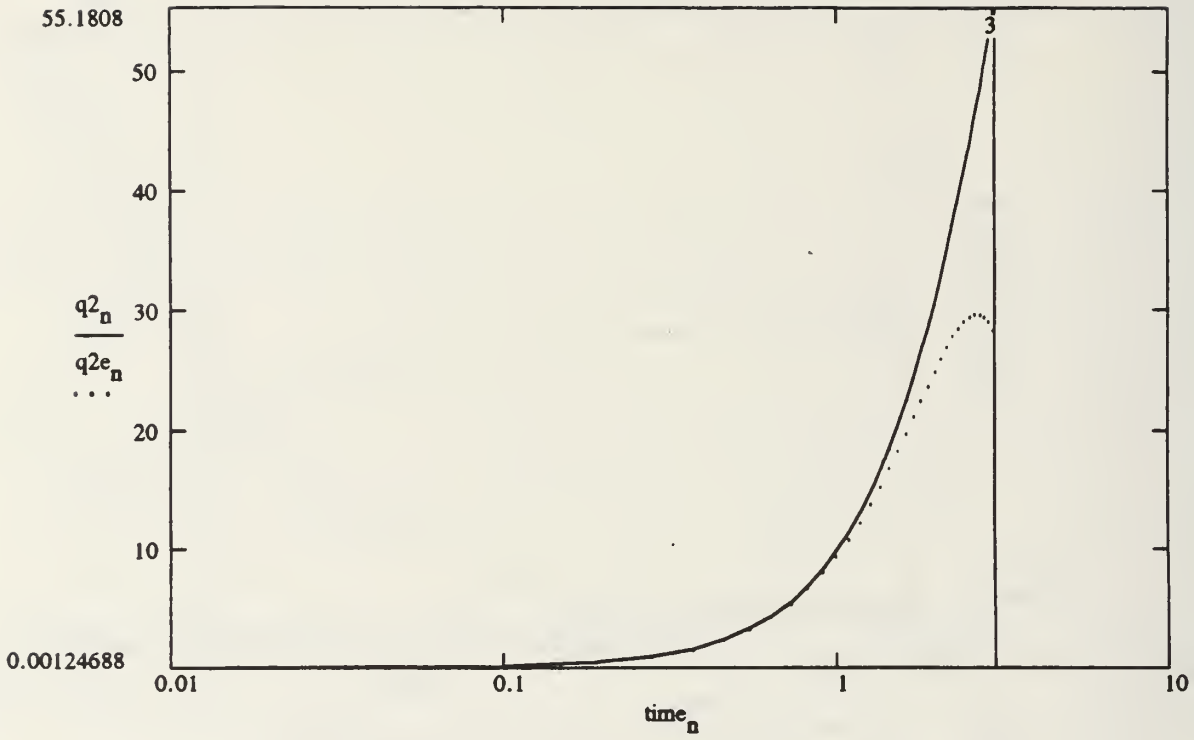

17584
15000
10000
$q1_t$
5000
8.31773e-006
0.01
0.1
1
10
100
time$_t$

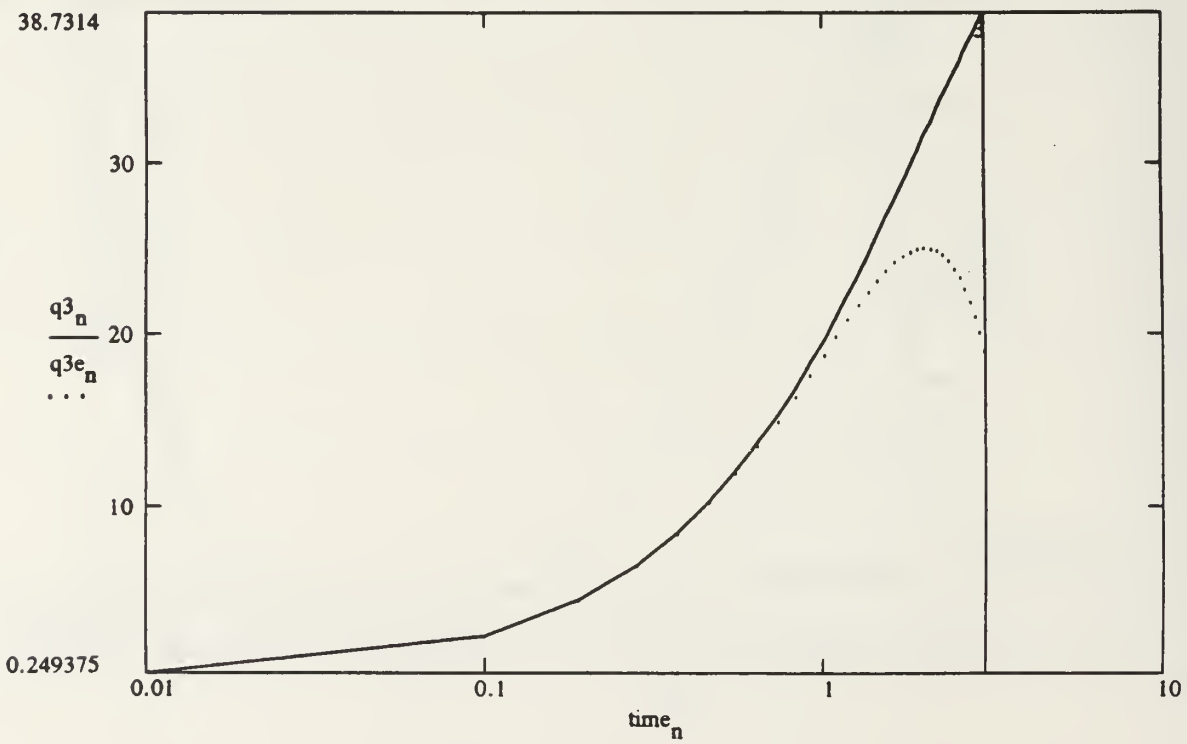

199.998
150
100
50
0.00124688
q2t
0.01
0.1
1
10
100
timet


50
40
30
20
10
0.249376
q3t
0.01
0.1
1
10
100
timet


2.10106
2
1.5
1
0.5
0.0099875
$\phi 2_n$
$\phi 3_n$
$\phi 3\varepsilon_n$
$\phi 2\varepsilon_n$
3
0.01
0.1
1
10
$time_n$


130.852
120
100
80
60
40
20
8.31771e-006
$q1_n$
$q1e_n$
0.01
0.1
1
10
$time_n$


55.1808
50
40
$q2_n$
$q2e_n$
30
20
10
0.00124688
3
0.01
0.1
1
10
$time_n$


38.7314
30
$q3_n$
$q3e_n$
20
10
0.249375
3
0.01
0.1
1
10
$time_n$

APPENDIX B: APPLICABLE SOURCE CODE

This appendix contains portions of the ASSET source code which would be modified to implement the changes described above. The remarks indicate the appropriate portions which would be deleted or changed. This is not an all inclusive listing, but contains the portions most affected by the modifications.


```
;---------------------------------------------------------------
;
;                          OBJECT ACT-MTST
;
;          ACT-MTST is an Object LISP implementation of MTST, the
;          Manuevering Target Statistical Tracker.
;
;---------------------------------------------------------------

(setq act-mtst (kindof nil))

;---------------------------------------------------------------

(defobfun (exist act-mtst) (init-list)
   (usual-exist init-list)
   (have 'dtor (/ pi 180))
   (have 'alpha 0.25)
   (have 'sigma (sqrt (/ 100 2)))       ; 10 knot target
)

;---------------------------------------------------------------

(defobfun (create-object-links act-mtst) (osis)
   (have 'ldbm (ask osis ldbm))
)

;---------------------------------------------------------------

(defobfun (start-new-track act-mtst) (uc track-id)
   (let* ((mean-cov-list (compute-mean-covariance uc))
          (mean (first  mean-cov-list))
          (var  (second mean-cov-list))
          (alt  (obu-contact-altitude uc))
          (uc-spatial-data (obu-contact-spatial-data uc))
          (lat  (obu-data-field-lat  uc-spatial-data))
          (lng  (obu-data-field-lng  uc-spatial-data))
          (ctc-list (list uc))
           track state-cov
         )
         (setq state-cov (make-obu-state-field
                          :ref-lat lat  :ref-lng    lng
                          :state   mean :covariance var))
         (return-from start-new-track
                      (make-obu-track :id track-id
                                      :number-of-contacts 1
                                      :head-state-covariance state-cov
                                      :head-contact-id (obu-contact-id uc)
                                      :head-spatial-data uc-spatial-data
                                      :tail-state-covariance state-cov
                                      :tail-contact-id (obu-contact-id uc)
                                      :tail-spatial-data uc-spatial-data
                                      :altitude alt
                                      :contacts ctc-list
                      )
         )
   )
```

```
)

;-------------------------------------------------------------------

(defobfun (compute-mean-covariance act-mtst) (uc)
   (let* ((uc-spatial-data (obu-contact-spatial-data uc))
          (type (obu-data-field-type uc-spatial-data))
          (lat  (obu-data-field-lat  uc-spatial-data))
          (lng  (obu-data-field-lng  uc-spatial-data))
          (brg  (obu-data-field-brg  uc-spatial-data))
          (maj  (obu-data-field-maj  uc-spatial-data))
          (min  (obu-data-field-min  uc-spatial-data))
          (cse  (obu-data-field-cse  uc-spatial-data))
          (cse-unc  (obu-data-field-cse-unc  uc-spatial-data))
          (spd  (obu-data-field-spd  uc-spatial-data))
          (spd-unc  (obu-data-field-spd-unc  uc-spatial-data))
          (mean (make-array 4      :initial-element 0))
          (var  (make-array '(4 4) :initial-element 0))
          stationary smaj smin cosb sinb cosb2 sinb2
         )
         (setq stationary (/ (* sigma sigma) (* 2 alpha)))
         (setf (aref mean 2) (* spd (cos (* dtor (- 90 cse)))))
         (setf (aref mean 3) (* spd (sin (* dtor (- 90 cse)))))
         (setf (aref var 2 2) stationary)
         (setf (aref var 3 3) stationary)
         (setq smaj (max maj 1))
         (setq smin (max min 1))
         (setq cosb (cos (* dtor brg)))
         (setq sinb (sin (* dtor brg)))
         (setq cosb2 (* cosb cosb))
         (setq sinb2 (* sinb sinb))
         (setq smaj2 (* 0.25 smaj smaj))
         (setq smin2 (* 0.25 smin smin))
         (setf (aref var 0 0) (+ (* smin2 cosb2) (* smaj2 sinb2)))
         (setf (aref var 0 1) (* (- smaj2 smin2) sinb cosb))
         (setf (aref var 1 0) (aref var 0 1))
         (setf (aref var 1 1) (+ (* smin2 sinb2) (* smaj2 cosb2)))
         (return-from compute-mean-covariance (list mean var))
    )
)

;-------------------------------------------------------------------

(defobfun (update-track act-mtst) (uc track)
   (let* ((uc-spatial-data (obu-contact-spatial-data uc))
          (id             (obu-contact-id           uc))
          (type (obu-data-field-type uc-spatial-data))
          (lat  (obu-data-field-lat  uc-spatial-data))
          (lng  (obu-data-field-lng  uc-spatial-data))
          (brg  (obu-data-field-brg  uc-spatial-data))
          (maj  (obu-data-field-maj  uc-spatial-data))
          (min  (obu-data-field-min  uc-spatial-data))
          (cse  (obu-data-field-cse  uc-spatial-data))
          (cse-unc  (obu-data-field-cse-unc uc-spatial-data))
          (spd  (obu-data-field-spd  uc-spatial-data))
          (spd-unc  (obu-data-field-spd-unc uc-spatial-data))
          (obs-time  (obu-data-field-obs-time uc-spatial-data))
```

```
          (track-spatial-data (obu-track-head-spatial-data track))
          (tail-spatial-data  (obu-track-tail-spatial-data track))
          (last-update-time (obu-data-field-obs-time track-spatial-data))
          (tail-update-time (obu-data-field-obs-time tail-spatial-data ))
          (last-state-cov (obu-track-head-state-covariance track))
          (tail-state-cov (obu-track-tail-state-covariance track))
          (last-ref-lat (obu-state-field-ref-lat last-state-cov))
          (last-ref-lng (obu-state-field-ref-lng last-state-cov))
           state-cov-old state-cov-ctc state-cov-new delta-time
           contacts time-old mean-var-list pos ctc ctc-sd count
         )
;
         (setq delta-time (- obs-time last-update-time))
;
         (if (>= delta-time 0)
           (progn
              (setq state-cov-old (change-tangent lat lng
                                  (ioumotion delta-time last-state-cov)))
              (setq state-cov-ctc (make-state type lat lng brg maj min cse cse-unc
                                           spd spd-unc))
              (setq state-cov-new (center-tangent (filter state-cov-old
state-cov-ctc)))
              (setq track (update-track-head track uc state-cov-new
uc-spatial-data id))
              (setq track (update-track-tail track))
            )
            (progn
              (if (or (> obs-time tail-update-time)
                      (< (obu-track-number-of-contacts track)
                         (ask ldbm contacts-from-head-to-tail)))
                (progn
                  (setq contacts (append (obu-track-contacts track) (list uc)))
                  (setq contacts (sort contacts #'(lambda (x y)
                        (< (obu-data-field-obs-time (obu-contact-spatial-data x))
                           (obu-data-field-obs-time (obu-contact-spatial-data
y))))))
                  (if (>= obs-time tail-update-time)
                    (progn
                      (setq state-cov-old tail-state-cov  )
                      (setq time-old      tail-update-time)
                    )
                    (progn
                      (setq mean-var-list (compute-mean-covariance (first
contacts)))
                      (setq lat (obu-data-field-lat (obu-contact-spatial-data
(first contacts))))
                      (setq lng (obu-data-field-lng (obu-contact-spatial-data
(first contacts))))
                      (setq state-cov-old (make-obu-state-field
                                  :ref-lat lat
                                  :ref-lng lng
                                  :state      (first  mean-var-list)
                                  :covariance (second mean-var-list)))
                      (setq time-old (obu-data-field-obs-time
                                       (obu-contact-spatial-data (first
contacts))))
                    )
```

```
                  )
                  (setq count (min (ask ldbm contacts-from-head-to-tail)
                                   (- (list-length contacts) 1)))
                  (dotimes (i count nil)
                     (setq pos (+ (obu-track-number-of-contacts track) (- count) 1
i))
                     (setq ctc      (nth pos contacts))
                     (setq ctc-sd   (obu-contact-spatial-data ctc))
                     (setq type     (obu-data-field-type     ctc-sd))
                     (setq obs-time (obu-data-field-obs-time ctc-sd))
                     (setq lat      (obu-data-field-lat      ctc-sd))
                     (setq lng      (obu-data-field-lng      ctc-sd))
                     (setq brg      (obu-data-field-brg      ctc-sd))
                     (setq maj      (obu-data-field-maj      ctc-sd))
                     (setq min      (obu-data-field-min      ctc-sd))
                     (setq cse      (obu-data-field-cse      ctc-sd))
                     (setq cse-unc  (obu-data-field-cse-unc  ctc-sd))
                     (setq spd      (obu-data-field-spd      ctc-sd))
                     (setq spd-unc  (obu-data-field-spd-unc  ctc-sd))
                     (setq delta-time (- obs-time time-old))
                     (setq state-cov-old (change-tangent lat lng
                                         (ioumotion delta-time state-cov-old)))
                     (setq state-cov-ctc (make-state type lat lng brg maj min cse
cse-unc
                                         spd spd-unc))
                     (setq state-cov-old (center-tangent
                                         (filter state-cov-old state-cov-ctc)))
                     (setq time-old obs-time)
                  )
                  (setq track (update-track-head track uc state-cov-old
                                                 (obu-contact-spatial-data (first
(last contacts)))
                                                 (obu-contact-id (first (last
contacts)))))
                  (setq track (update-track-tail track))
               )
            )
         )
      )
;
      (return-from update-track track)
   )
)

;---------------------------------------------------------------------

(defobfun (update-track-head act-mtst) (track uc state-cov spatial-data id)
 (let ((contacts (append (obu-track-contacts track) (list uc)))
      )
   (setq contacts (sort contacts #'(lambda (x y)
                        (< (obu-data-field-obs-time (obu-contact-spatial-data x))
                           (obu-data-field-obs-time (obu-contact-spatial-data
y))))))
   (setf (obu-track-number-of-contacts track) (+ (obu-track-number-of-contacts
track) 1))
   (setf (obu-track-head-state-covariance track) state-cov   )
   (setf (obu-track-head-spatial-data     track) spatial-data)
```

```
   (setf (obu-track-head-contact-id       track) id          )
   (setf (obu-track-contacts              track) contacts    )
   (return-from update-track-head track)
 )
)

;-----------------------------------------------------------------

(defobfun (update-track-tail act-mtst) (track)
   (let* ((nctcs (obu-track-number-of-contacts track))
          (head-sc (obu-track-head-state-covariance track))
          (head-sd (obu-track-head-spatial-data     track))
          (head-id (obu-track-head-contact-id       track))
          (tail-sc (obu-track-tail-state-covariance track))
          (tail-sd (obu-track-tail-spatial-data     track))
          (tail-id (obu-track-tail-contact-id       track))
          (tail-update-time (obu-data-field-obs-time tail-sd))
          (contacts (obu-track-contacts track))
           type obs-time lat lng brg maj min cse cse-unc spd spd-unc
           mean-var-list new-tail pos-new-tail new-tail-sd delta-time
           state-cov-old state-cov-ctc state-cov-new
         )
         (setq contacts (sort contacts #'(lambda (x y)
                        (< (obu-data-field-obs-time (obu-contact-spatial-data x))
                           (obu-data-field-obs-time (obu-contact-spatial-data
y)))))
         )
         (if (<= nctcs 1)
           (progn (setf (obu-track-tail-state-covariance track) head-sc)
                  (setf (obu-track-tail-spatial-data     track) head-sd)
                  (setf (obu-track-tail-contact-id       track) head-id)
                  (return-from update-track-tail track)
           )
         )

         (if (<= nctcs (ask ldbm contacts-from-head-to-tail))
           (if (= tail-id (obu-contact-id (first contacts)))
             (return-from update-track-tail track)
             (progn
               (setq mean-var-list (compute-mean-covariance (first contacts)))
               (setq lat (obu-data-field-lat (obu-contact-spatial-data (first
contacts))))
               (setq lng (obu-data-field-lng (obu-contact-spatial-data (first
contacts))))
               (setf (obu-track-tail-state-covariance track) (make-obu-state-field
                                  :ref-lat lat
                                  :ref-lng lng
                                  :state      (first  mean-var-list)
                                  :covariance (second mean-var-list)))
               (setf (obu-track-tail-spatial-data track) (obu-contact-spatial-data
                                  (first contacts)))
               (setf (obu-track-tail-contact-id track) (obu-contact-id (first
contacts)))
               (return-from update-track-tail track)
             )
           )
         )
```

```
          (setq pos-new-tail (- (obu-track-number-of-contacts track)
                                (ask ldbm contacts-from-head-to-tail)))
          (setq new-tail (nth pos-new-tail (obu-track-contacts track)))
          (setq new-tail-sd (obu-contact-spatial-data new-tail))
          (setq type      (obu-data-field-type     new-tail-sd))
          (setq obs-time (obu-data-field-obs-time new-tail-sd))
          (setq lat       (obu-data-field-lat      new-tail-sd))
          (setq lng       (obu-data-field-lng      new-tail-sd))
          (setq brg       (obu-data-field-brg      new-tail-sd))
          (setq maj       (obu-data-field-maj      new-tail-sd))
          (setq min       (obu-data-field-min      new-tail-sd))
          (setq cse       (obu-data-field-cse      new-tail-sd))
          (setq cse-unc   (obu-data-field-cse-unc  new-tail-sd))
          (setq spd       (obu-data-field-spd      new-tail-sd))
          (setq spd-unc   (obu-data-field-spd-unc  new-tail-sd))
          (setq delta-time (- obs-time tail-update-time))
          (setq state-cov-old (change-tangent lat lng (ioumotion delta-time
tail-sc)))
          (setq state-cov-ctc (make-state type lat lng brg maj min cse cse-unc spd
spd-unc))
          (setq state-cov-new (center-tangent (filter state-cov-old
state-cov-ctc)))
          (setf (obu-track-tail-state-covariance track) state-cov-new)
          (setf (obu-track-tail-spatial-data     track) new-tail-sd  )
          (setf (obu-track-tail-contact-id       track) (obu-contact-id new-tail))

          (prune-contacts-before-tail track)  ; Get rid of contacts before tail.

          (return-from update-track-tail track)
    )
)


;-----------------------------------------------------------------

(defobfun (prune-contacts-before-tail act-mtst) (track)
   (let ((nctcs     (obu-track-number-of-contacts track))
         (contacts (obu-track-contacts           track))
         (max-ctcs (ask ldbm contacts-from-head-to-tail))
          tail-position
        )
        (setq tail-position (- nctcs max-ctcs))
        (if (<= tail-position 0)
            (return-from prune-contacts-before-tail nil)
        )
        (dotimes (i tail-position nil)
           (setq contacts (remove-element contacts 0))
        )
        (setf (obu-track-number-of-contacts track) max-ctcs)
        (setf (obu-track-contacts           track) contacts)
   )
)

;-----------------------------------------------------------------

(defobfun (ioumotion act-mtst) (time state-cov)
```

```
    (let ((lat  (obu-state-field-ref-lat    state-cov))
          (lng  (obu-state-field-ref-lng    state-cov))
          (mean (obu-state-field-state      state-cov))
          (var  (obu-state-field-covariance state-cov))
           motion-matrices phi f new-mean new-var
           vartmp phitrn
         )
         (if (= time 0)
             (return-from ioumotion state-cov)
         )
         (setq motion-matrices (calc-mot-mat time))
         (setq phi (first  motion-matrices))
         (setq f   (second motion-matrices))
         (setq new-mean (matrix-multiply phi mean))
         (setq vartmp   (matrix-multiply phi var ))
         (setq phitrn   (matrix-transpose phi))
         (setq new-var (matrix-add (matrix-multiply vartmp phitrn) f))
         (setq phi new-var)
         (setq phitrn (matrix-transpose new-var))
         (setq new-var (matrix-scalar-multiply 0.5 (matrix-add phi phitrn)))
         (return-from ioumotion (make-obu-state-field
                                  :ref-lat lat
                                  :ref-lng lng
                                  :state new-mean
                                  :covariance new-var)
         )
    )
)

;-----------------------------------------------------------------

(defobfun (calc-mot-mat act-mtst) (time)
   (let* ((invalpha (/ 1 alpha))
          (invalpha2 (* invalpha invalpha))
          (expalphat (exponential (* -1 alpha time)))
          (exp2alphat (* expalphat expalphat))
          (f   (make-array '(4 4) :initial-element 0))
          (phi (make-array '(4 4) :initial-element 0))
         )
         (if (> time 0)
           (progn
              (setf (aref f 0 0) (* invalpha2 (- time (* invalpha
                    (- (* 2 (- 1 expalphat)) (* .5 (- 1 exp2alphat)))))))
              (setf (aref f 1 1) (aref f 0 0))
              (setf (aref f 0 2) (* invalpha2 (+ 0.5 (- expalphat)
                    (* .5 exp2alphat))))
              (setf (aref f 2 0) (aref f 0 2))
              (setf (aref f 1 3) (aref f 0 2))
              (setf (aref f 3 1) (aref f 0 2))
              (setf (aref f 2 2) (* 0.5 invalpha (- 1 exp2alphat)))
              (setf (aref f 3 3) (aref f 2 2))
           )
           (progn
              (setf (aref f 0 0) (* invalpha2 (- (- time) (* invalpha
                    (- (* 2 (- expalphat 1)) (* .5 (- exp2alphat 1)))))))
              (setf (aref f 1 1) (aref f 0 0))
              (setf (aref f 0 2) (* invalpha2 (+ (- 0.5) expalphat
```

```
                    (* -1 .5 exp2alphat))))
                (setf (aref f 2 0) (aref f 0 2))
                (setf (aref f 1 3) (aref f 0 2))
                (setf (aref f 3 1) (aref f 0 2))
                (setf (aref f 2 2) (* 0.5 invalpha (- exp2alphat 1)))
                (setf (aref f 3 3) (aref f 2 2))
              )
            )
            (setq f (matrix-scalar-multiply (* sigma sigma) f))
            (setq phi (identity-matrix 4))
            (setf (aref phi 0 2) (* invalpha (- 1 expalphat)))
            (setf (aref phi 1 3) (aref phi 0 2))
            (setf (aref phi 2 2) expalphat)
            (setf (aref phi 3 3) (aref phi 2 2))
            (return-from calc-mot-mat (list phi f))
      )
)

;-----------------------------------------------------------------

(defobfun (exponential act-mtst) (x)
      (if (< x -350)
          (return-from exponential (exp -350))
      )
      (if (> x 350)
          (return-from exponential (exp 350))
          (return-from exponential (exp x))
      )
)

;-----------------------------------------------------------------

(defobfun (make-state act-mtst) (type lat lng brg maj min cse cse-unc spd spd-unc)
      (let ((mean (make-array   4    :initial-element 0))
            (var  (make-array '(4 4) :initial-element 0))
            (stationary (/ (* sigma sigma) (* 2 alpha)))
            (smaj (max maj 1)) (smin (max min 1))
           )
           (setf (aref mean 2) (* spd (cos (* dtor (- 90 cse)))))
           (setf (aref mean 3) (* spd (sin (* dtor (- 90 cse)))))
           (setq var (make-var stationary brg smaj smin))
           (return-from make-state (make-obu-state-field
                                    :ref-lat lat
                                    :ref-lng lng
                                    :state mean
                                    :covariance var))
      )
)

;-----------------------------------------------------------------

(defobfun (make-var act-mtst) (stationary brg smaj smin)
      (let* ((var (make-array '(4 4) :initial-element 0))
             (cosb (cos (* dtor brg)))
             (sinb (sin (* dtor brg)))
             (cosb2 (* cosb cosb))
             (sinb2 (* sinb sinb))
```

```
          (smaj2 (* 0.25 smaj smaj))
          (smin2 (* 0.25 smin smin))
         )
         (setf (aref var 0 0) (+ (* smin2 cosb2) (* smaj2 sinb2)))
         (setf (aref var 0 1) (* (- smaj2 smin2) sinb cosb))
         (setf (aref var 1 0) (aref var 0 1))
         (setf (aref var 1 1) (+ (* smin2 sinb2) (* smaj2 cosb2)))
         (setf (aref var 2 2) stationary)
         (setf (aref var 3 3) stationary)
         (return-from make-var var)
   )
)

;-----------------------------------------------------------------

(defobfun (filter act-mtst) (state-cov-sol state-cov-ctc)
   (let ((lat    (obu-state-field-ref-lat    state-cov-sol))
         (lng    (obu-state-field-ref-lng    state-cov-sol))
         (mean1  (obu-state-field-state      state-cov-sol))
         (var1   (obu-state-field-covariance state-cov-sol))
         (mean2  (obu-state-field-state      state-cov-ctc))
         (var2   (obu-state-field-covariance state-cov-ctc))
         (infmat1 (make-array '(4 4)))
         (infmat2 (make-array '(4 4)))
         (infvec1 (make-array 4))
         (infvec2 (make-array 4))
         (tmpmat  (make-array '(4 4)))
         (tmpvec  (make-array 4))
          mean var
        )
        (setq infmat1 (matrix-invert var1))
        (setq infvec1 (matrix-multiply infmat1 mean1))
        (setq infmat2 (matrix-invert var2))
        (setq infvec2 (matrix-multiply infmat2 mean2))
        (setq tmpmat (matrix-add infmat1 infmat2))
        (setq var (matrix-invert tmpmat))
        (setq tmpvec (matrix-add infvec1 infvec2))
        (setq mean (matrix-multiply var tmpvec))
        (return-from filter (make-obu-state-field
                             :ref-lat lat
                             :ref-lng lng
                             :state mean
                             :covariance var)
        )
   )
)

;-----------------------------------------------------------------

(defobfun (change-tangent act-mtst) (newlat newlng state-cov)
   (let ((oldlat (obu-state-field-ref-lat    state-cov))
         (oldlng (obu-state-field-ref-lng    state-cov))
         (mean   (obu-state-field-state      state-cov))
         (var    (obu-state-field-covariance state-cov))
         (rot    (make-array '(4 4) :initial-element 0))
         (rottrn (make-array '(4 4)))
         (newtmp (make-array '(4 4)))
```

```
         (new-mean (make-array  4    ))
         (new-var  (make-array '(4 4)))
         latlnglist rngbrglist cenlat cenlng deltabrg cosb sinb
         rngl rng2 rng3 brgl brg2 brg3
        )
        (setq rngl (sqrt (+ (* (aref mean 0) (aref mean 0))
                            (* (aref mean 1) (aref mean 1)))))
        (setq brgl (arctan (aref mean 1) (aref mean 0)))
        (setq latlnglist (getlatlng oldlat oldlng rngl brgl 0))
        (setq cenlat (first  latlnglist))
        (setq cenlng (second latlnglist))
        (setq brgl (second (getrngbrg oldlat oldlng cenlat cenlng 0)))
        (setq brg2 (second (getrngbrg cenlat cenlng oldlat oldlng 0)))
        (setq rngbrglist (getrngbrg newlat newlng cenlat cenlng 0))
        (setq rng3 (first  rngbrglist))
        (setq brg3 (second rngbrglist))
        (setq brg4 (second (getrngbrg cenlat cenlng newlat newlng 0)))
        (setq deltabrg (mod (+ (- brg2 brgl) (- brg3 brg4) 360) 360))
        (setq cosb (cos (* dtor deltabrg)))
        (setq sinb (sin (* dtor deltabrg)))
;
;             Move the mean position:
        (setf (aref new-mean 0) (* rng3 (cos (* dtor (- 90 brg3)))))
        (setf (aref new-mean 1) (* rng3 (sin (* dtor (- 90 brg3)))))
;
;             Rotate the velocity:
        (setf (aref new-mean 2) (+ (* cosb (aref mean 2)) (* sinb (aref mean 3))))
        (setf (aref new-mean 3) (- (* cosb (aref mean 3)) (* sinb (aref mean 2))))
;
;             Rotate the covariance:
        (setf (aref rot 0 0) cosb)
        (setf (aref rot 1 1) cosb)
        (setf (aref rot 2 2) cosb)
        (setf (aref rot 3 3) cosb)
        (setf (aref rot 0 1) sinb)
        (setf (aref rot 2 3) sinb)
        (setf (aref rot 1 0) (- sinb))
        (setf (aref rot 3 2) (- sinb))
;
        (setq newtmp (matrix-multiply rot var))
        (setq rottrn (matrix-transpose rot))
        (setq new-var (matrix-multiply newtmp rottrn))
;
        (setq rot new-var)
        (setq rottrn (matrix-transpose new-var))
        (setq new-var (matrix-scalar-multiply 0.5 (matrix-add rot rottrn)))
        (return-from change-tangent (make-obu-state-field
                                     :ref-lat    newlat
                                     :ref-lng    newlng
                                     :state      new-mean
                                     :covariance new-var)
        )
    )
)

;-----------------------------------------------------------------
```

```
(defobfun (center-tangent act-mtst) (state-cov)
   (let ((lat  (obu-state-field-ref-lat    state-cov))
         (lng  (obu-state-field-ref-lng    state-cov))
         (mean (obu-state-field-state      state-cov))
         (var  (obu-state-field-covariance state-cov))
         (rot  (make-array '(4 4) :initial-element 0))
         (rottrn (make-array '(4 4)))
         (newtmp (make-array '(4 4)))
         (new-mean (make-array   4   ))
         (new-var  (make-array '(4 4)))
         latlnglist rngbrglist newlat newlng newrng newbrg
         rngl brgl cosb sinb
        )
        (setq rngl (sqrt (+ (* (aref mean 0) (aref mean 0))
                            (* (aref mean 1) (aref mean 1)))))
        (if (< rngl 0.01)
            (return-from center-tangent state-cov)
        )
        (setq brgl (arctan (aref mean 1) (aref mean 0)))
        (setq latlnglist (getlatlng lat lng rngl brgl 0))
        (setq newlat (first  latlnglist))
        (setq newlng (second latlnglist))
        (setq rngbrglist (getrngbrg newlat newlng lat lng 0))
        (setq newrng (first  rngbrglist))
        (setq newbrg (mod (+ (second rngbrglist) 180) 360))
        (setq deltabrg (- newbrg brgl))
        (setq cosb (cos (* dtor deltabrg)))
        (setq sinb (sin (* dtor deltabrg)))
;
;            Move the mean position:
        (setf (aref new-mean 0) 0)
        (setf (aref new-mean 1) 0)
;
;            Rotate the velocity:
        (setf (aref new-mean 2) (+ (* cosb (aref mean 2)) (* sinb (aref mean 3))))
        (setf (aref new-mean 3) (- (* cosb (aref mean 3)) (* sinb (aref mean 2))))
;
;            Rotate the covariance:
        (setf (aref rot 0 0) cosb)
        (setf (aref rot 1 1) cosb)
        (setf (aref rot 2 2) cosb)
        (setf (aref rot 3 3) cosb)
        (setf (aref rot 0 1) sinb)
        (setf (aref rot 2 3) sinb)
        (setf (aref rot 1 0) (- sinb))
        (setf (aref rot 3 2) (- sinb))
;
        (setq newtmp (matrix-multiply rot var))
        (setq rottrn (matrix-transpose rot))
        (setq new-var (matrix-multiply newtmp rottrn))
;
        (setq rot new-var)
        (setq rottrn (matrix-transpose new-var))
        (setq new-var (matrix-scalar-multiply 0.5 (matrix-add rot rottrn)))
        (return-from center-tangent (make-obu-state-field
                                     :ref-lat    newlat
                                     :ref-lng    newlng
```

```
                                        :state      new-mean
                                        :covariance new-var)
        )
    )
)

;-----------------------------------------------------------------

(defobfun (arctan act-mtst) (y x)
   (let (a
         )
         (if (and (= y 0) (= x 0))
             (return-from arctan 0)
         )
         (if (= y 0)
             (progn (if (< x 0)
                        (return-from arctan 270)
                        (return-from arctan  90)
                    )
             )
         )
         (if (< y 0)
             (setq a (+ (/ (atan (/ x y)) dtor) 180))
             (setq a    (/ (atan (/ x y)) dtor)     )
         )
         (if (> a 0)
             (return-from arctan a)
             (return-from arctan (+ a 360))
         )
    )
)

;-----------------------------------------------------------------
;
;     This procedure converts a covariance matrix (cov) into an
;     ellipse.  The ellipse is returned in the form:
;
;           ( Orientation , Semi-major axis , Semi-minor axis ).
;
;-----------------------------------------------------------------


(defobfun (cov-to-ellipse act-mtst) (cov)
   (let ((a (aref cov 0 0))
         (b (aref cov 0 1))
         (c (aref cov 1 1))
          brg cosb sinb cosb2 sinb2 sincos2b smjr smnr
        )
        (if (not (> (- (* a c) (* b b)) 0))
            (return-from cov-to-ellipse (list 0 (expt 10 9) (expt 10 9)))
        )
        (if (= b 0)
            (if (not (> a c))
                (return-from cov-to-ellipse (list 0  (* 2 (sqrt (abs c)))
                                                     (* 2 (sqrt (abs a)))))
                (return-from cov-to-ellipse (list 90 (* 2 (sqrt (abs a)))
                                                     (* 2 (sqrt (abs c)))))


            )
        )
        (if (> b 0)
            (if (= a c)
                (return-from cov-to-ellipse (list 45 (* 2 (sqrt (+ a b)))
                                                     (* 2 (sqrt (- a b)))))
            )
        )
        (if (< b 0)
            (if (= a c)
                (return-from cov-to-ellipse (list 135 (* 2 (sqrt (- a b)))
                                                      (* 2 (sqrt (+ a b)))))
            )
        )
        (setq brg (/ (arctan (- c a) (* 2 b)) 2))
        (setq sinb (sin (* dtor brg)))
        (setq cosb (cos (* dtor brg)))
        (setq sinb2 (* sinb sinb))
        (setq cosb2 (* cosb cosb))
        (setq sincos2b (* 2 b sinb cosb))
        (setq smjr (* 2 (sqrt (+ (* a sinb2) sincos2b (* c cosb2)))))
        (setq smnr (* 2 (sqrt (+ (* a cosb2) (* -1 sincos2b) (* c sinb2)))))
        (return-from cov-to-ellipse (list brg smjr smnr))
   )
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;lisp matrix manipulation routines
;4/28/89--s.c. lent
;
; these routines operate on three(3) 'standard type' lisp objects which are
; subsets of lisp object types:
;
;            1. scalars, i.e., lisp type 'number'
;            2. vectors, i.e., lisp type 'simple-vector' with
;                              length > 1
;            3. matrices, i.e., lisp type 'array' with
;                               rank = 2 and
;                               each dimension > 1
;
; all routines accept input, and produce output of these types
;
; unacceptable inputs produce 'NIL' as output.  there are two(2) cases:
;            1. 'non-standard' inputs
;            2. 'standard' but mismatched inputs, e.g.,
;               the inverse of matrix(m x n), where m <> n.
;
;
;routines consist of:
;            1. mx-mult
;            2. mx-add
;            3. sc-mx-mult
;            4. mx-trans
;            5. mx-ident
;            6. mx-inverse
;
;
;            these are built using the following 'sub-routines':
;
;                  1. matrixp
;                  2. array-row-el
;                  3. array-col-el
;                  4. array-row-dim
;                  5. array-col-dim
;                  6. can-mx-mult
;                  7. can-mx-add
;                  8. sqr-mx
;                  9. mx-factor
;                 10. mx-subst
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defun matrix-multiply (a b)
 (if (not (can-mx-mult a b))
     (return-from matrix-multiply NIL)
 )
 (let ((c) (ij-list))
  (setq c (make-array (remove 1 (list (array-row-dim a)(array-col-dim b)))
          )
```

```
  )
  (do ((index1 0 (+ index1 1)))
      ((> index1 (- (array-col-dim b) 1)))
      (if (> (array-col-dim b) 1) (setq ij-list (list index1))
                                  (setq ij-list ()))
      )
      (do ((index2 0 (+ index2 1)))
          ((> index2 (- (array-row-dim a) 1)))
          (if (> (array-row-dim a) 1) (setq ij-list (cons  index2 ij-list))
          )
          (setf (apply #'aref c ij-list) 0)
          (do ((index3 0 (+ index3 1)))
              ((> index3 (- (array-dimension b 0) 1)))
              (setf (apply #'aref c ij-list) (+ (apply #'aref c ij-list)
                                                (* (array-col-el a (list index2
index3))
                                                   (array-row-el b (list index3
index1))
                                                )
                                                )
              )
          )
          (if (> (array-row-dim a) 1) (setq ij-list (cdr ij-list))
          )
      )
  )
  (if (equal (array-dimensions c) () )
      (aref c)
      c
  )
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun matrix-scalar-multiply (sc mx)
 (if (not (and (numberp sc)(matrixp mx)))
     (return-from matrix-scalar-multiply NIL)
 )
 (let ((ij-list)(mx-new (make-array (array-dimensions mx))))
  (do ((index1 0 (+ index1 1)))
      ((> index1 (- (array-col-dim mx) 1)))
      (if (> (array-col-dim mx) 1) (setq ij-list (list index1))
                                   (setq ij-list ()))
      )
      (do ((index2 0 (+ index2 1)))
          ((> index2 (- (array-dimension mx 0) 1)))
          (if (> (array-dimension mx 0) 1) (setq ij-list (cons  index2 ij-list))
          )
          (setf (apply #'aref mx-new ij-list) (* (apply #'aref mx ij-list) sc))
          (setq ij-list (cdr ij-list))
      )
  )
  mx-new
 )
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun matrix-transpose (mx)
 (if (not (matrixp mx))
     (return-from matrix-transpose NIL)
 )
 (let ((ij-list) (mx-new (make-array (reverse (array-dimensions mx)))))
  (do ((index1 0 (+ index1 1)))
      ((> index1 (- (array-col-dim mx) 1)))
      (if (> (array-col-dim mx) 1) (setq ij-list (list index1))
                                   (setq ij-list ())
      )
      (do ((index2 0 (+ index2 1)))
          ((> index2 (- (array-dimension mx 0) 1)))
          (if (> (array-dimension mx 0) 1) (setq ij-list (cons  index2 ij-list))
          )
          (setf (apply #'aref mx-new (reverse ij-list)) (apply #'aref mx ij-list))
          (setq ij-list (cdr ij-list))
      )
  )
  mx-new
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun matrix-add (mx1 mx2)
 (if (not (can-mx-add mx1 mx2))
     (return-from matrix-add NIL)
 )
 (let ((ij-list)(mx-new (make-array (array-dimensions mx1))))
  (do ((index1 0 (+ index1 1)))
      ((> index1 (- (array-col-dim mx1) 1)))
      (if (> (array-col-dim mx1) 1) (setq ij-list (list index1))
                                    (setq ij-list ())
      )
      (do ((index2 0 (+ index2 1)))
          ((> index2 (- (array-dimension mx1 0) 1)))
          (if (> (array-dimension mx1 0) 1) (setq ij-list (cons  index2 ij-list))
          )
          (setf (apply #'aref mx-new ij-list) (+ (apply #'aref mx1 ij-list)
                                                 (apply #'aref mx2 ij-list)))
          (setq ij-list (cdr ij-list))
      )
  )
  mx-new
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun identity-matrix (m)
 (if (not (and (integerp m)(> m 1)))
     (return-from identity-matrix NIL)
 )
 (let ((ij-list)(mx-new (make-array (list m m))))
  (do ((index1 0 (+ index1 1)))
```

```
      ((> index1 (- m 1)))
      (setq ij-list (list index1))
      (do ((index2 0 (+ index2 1)))
          ((> index2 (- m 1)))
          (setq ij-list (cons  index2 ij-list))
          (setf (apply #'aref mx-new ij-list) (if (= index1 index2) 1 0))
          (setq ij-list (cdr ij-list))
      )
  )
  mx-new
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun matrix-invert (a)
  (if (not (sqr-mx a))
      (return-from matrix-invert NIL)
  )
  (let* ((n (array-dimension a 0))
         (temp   (mx-factor a n))
         (w      (car  temp))
         (ipivot (cadr temp))
        )
        (if (= 0 (caddr temp))
            (return-from matrix-invert NIL)
        )
        (mx-subst w n ipivot)
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun matrixp (a)
   (if (arrayp a)
       (cond ((simple-vector-p a )
              (if (> (array-dimension a 0) 0)
                  T
                  NIL
              )
             )
             ((equal(array-rank a) 2)
              (if (and (> (array-dimension a 0) 0)
                       (> (array-dimension a 1) 0)
                  )
                  T
                  NIL
              )
             )
             (T NIL)
       )
       NIL
   )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(defun array-row-el  (a i)
    (apply #'aref a (if (= (array-rank a) 1)
                        (list (car i))
                        i
                    )
    )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun array-col-el  (a i)
   (apply #'aref a (if (= (array-rank a) 1)
                       (cdr i)
                       i
                   )
   )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun array-row-dim (a)
   (if (= (array-rank a) 1)
       1
       (array-dimension a 0)
   )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun array-col-dim (a)
   (if (= (array-rank a) 1)
       1
       (array-dimension a (- (array-rank a) 1))
   )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun can-mx-mult (a b)
   (if (and (matrixp a)(matrixp b))
       (if (= (array-dimension a (- (array-rank a) 1))
              (array-dimension b 0)
           )
           T
           NIL
       )
       NIL
   )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun can-mx-add (a b)
   (if (and (matrixp a)(matrixp b))
       (if (and (= (array-dimension a 0)
                   (array-dimension b 0)
               )
```

```
                (= (array-dimension a (- (array-rank a) 1))
                   (array-dimension b (- (array-rank b) 1))
                )
            )
            T
            NIL
        )
        NIL
    )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun sqr-mx (a)
   (if (matrixp a)
       (if (= (array-rank a) 2)
           (if (= (array-dimension a 0)(array-dimension a 1))
           T
           NIL
           )
       )
       NIL
   )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun mx-factor (mx n)
 (let ((awikod)(colmax)(ratio)(rowmax)(temp) (istar)(iflag 1)
       (ipivot (make-array n))
       (d      (make-array n))
       (w      (make-array (array-dimensions mx)))
      )
      (if (<= n 0)
          (return-from mx-factor nil)
      )
      ;;;;;;;;;;;
      (do ((i 0 (+ i 1))
           (rowmax 0 0)
          )
          ((= i n))
          (setf (aref ipivot i) i)
          (do ((j 0 (+ j 1)))
              ((= j n))
              (setf (aref w i j)(aref mx i j))
              (setq rowmax (max rowmax (abs (aref  w i j))))
          )
          (when (= rowmax 0)
                (setq iflag 0)
                (setq rowmax 1)
          )
          (setf (aref d i) rowmax)
      )
      ;;;;;;;;;;;
      (do ((k 0 (+ k 1)))
          ((= k (- n 1)))
          (setq colmax (abs (/ (aref w k k)(aref d k))))
```

```
          (setq istar k)
          (do (( i (+ k 1) (+ i 1)))
              ((= i n))
              (setq awikod (abs (/ (aref w i k)(aref d i))))
              (when (> awikod colmax)
                    (setq colmax awikod)
                    (setq istar i)
              )
          )
          (if (= colmax 0)
              (setq iflag 0)
              (progn
                 (if (> istar k)
                     (progn
                       (setq iflag (* -1 iflag))
                       (setq i (aref ipivot istar))
                       (setf (aref ipivot istar) (aref ipivot k))
                       (setf (aref ipivot k) i)
                       (setq temp (aref d istar))
                       (setf (aref d istar) (aref d k))
                       (setf (aref d k) temp)
                       (do ((j 0 (+ j 1)))
                           ((= j n))
                           (setq temp (aref w istar j))
                           (setf (aref w istar j)(aref w k j))
                           (setf (aref w k j) temp)
                       )
                     )
                 )
                 (do ((i (+ k 1) (+ i 1)))
                     ((= i n))
                     (setf (aref w i k) (/ (aref w i k) (aref w k k)))
                     (setq ratio (aref w i k))
                     (do (( j (+ k 1)(+ j 1)))
                         ((= j n))
                         (setf (aref w i j) (- (aref w i j) (* ratio (aref w k
j))))
                     )
                 )
              )
          )
      )
      (if (= (aref w (- n 1)(- n 1)) 0)
          (setq iflag 0)
      )
      (list w ipivot iflag)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun mx-subst (w n ipivot)
 (let (
       (b      (make-array n :initial-element 0))
       (ainv   (make-array (list n n)))
       (ip)
      )

       (do ((col 0 (+ col 1)))
           ((= col n))
           (setf (aref b col) 1)
       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
           (setq ip (aref ipivot 0))
           (setf (aref ainv  0 col)(aref b ip))
           ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
           (do ((i 1 (+ i 1))
                (sum 0 0)
               )
               ((= i n))
               (do ((j 0 (+ j 1)))
                   ((= j i))
                   (setq sum (+ sum (* (aref w i j)(aref ainv j col))))
               )
               (setq ip (aref ipivot i))
               (setf (aref ainv i col ) (- (aref b ip) sum))
           )
           (setf (aref ainv (- n 1) col) (/ (aref ainv (- n 1) col) (aref w (- n
1)(- n 1))))
           ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
           (do (( i (- n 2) (- i 1))
                ( sum 0 0)
               )
               ((= i -1))
               (do (( j (+ i 1) (+ j 1)))
                   ((= j n))
                   (setq sum (+ sum (* (aref ainv j col)(aref w i j))))
               )
               (setf (aref ainv i col) (/ (- (aref ainv i col) sum)(aref w i i)))
           )
       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
           (setf (aref b col) 0)
       )
       ainv
 )
)
```

```
;;..... TRACK Utility Procedures:


(defun getLatLng (lat1 lng1 dist brg flag)
  ;flag = 0 great circle calc
  ;     = 1 rhumbline
  ; brg in degrees.
  (let* ((lat2 lat1) (lng2 lng1) (rng (rem (/ dist 60.1077) 360))
         (cbrg (cosLisp brg)) (lat1t lat1) (clat1t) (slat1)
         (srng nil) (crng nil) (dlng nil) (temp nil) (clat2 nil)
        )
    (if (= lat1 90) (setq lat1t 89.9))
    (if (= lat1 -90) (setq lat1t -89.9))
    (setq clat1 (cosLisp lat1t))
    (if (= flag 0)      ; great circle calculation
      (progn (if (> rng 180)
                 (progn (setq rng (- 360 rng))
                        (setq brg (rem (+ 180 brg) 360))
                        (setq cbrg (cosLisp brg))
                 )
               )
               (if (= clat1 0)
                 (progn (if (= (sinLisp brg) 1)
                          (setq lat2 (- lat1 rng))
                          (setq lat2 (+ lat1 rng))
                        )
                        (setq lng2 lng1)
                 )
                 (progn (setq slat1 (sinLisp lat1t))
                        (setq srng (sinLisp rng))
                        (setq crng (cosLisp rng))
                        (setq lat2 (- 90
                                      (acosLisp (fnAbs (+ (* clat1 srng cbrg)
                                                          (* slat1 crng))))))
                        (setq clat2 (cosLisp lat2))
                        (setq dlng (acosLisp (fnAbs (/ (- crng (* slat1
                                                                  (sinLisp lat2)))
                                                       (* clat1 clat2))))))
                        (if (< (sinLisp brg) 0)
                          (setq lng2 (fnCor (- lng1 dlng)))
                          (setq lng2 (fnCor (+ lng1 dlng)))
                        )
                 )
               )                        ;if clat1=0
      )                                 ; progn rng > 180
    )                                   ;if flag = gc
    (if (= flag 1)   ;rhumbline calc
      (progn (if (or (= brg 90)
                     (= brg 270))
                 (progn (setq lat2 lat1t)
                        (if (= brg 270)
                          (setq lng2 (fnCor (- lng1
                                               (/ rng clat1))))
                          (setq lng2 (fnCor (+ lng1

                                               (/ rng clat1))))
                        )
                 ) ;end then construct brg=90 or brg=270
                 (progn (setq lat2 (+ lat1t (* rng cbrg)))
                        (if (>= (abs lat2) 90)
                          (progn (if (> lat2 0)
                                   (setq lat2 90)
                                   (setq lat2 -90)
                                 )
                                 (setq lng2 0)
                          )
                          (progn (setq temp (log (/ (tanLisp (- 45 (* 0.5 lat1t)))
                                                    (tanLisp (- 45 (* 0.5 lat2))))))
                                 (setq lng2 (fnCor (+ lng1 (* GLB.radToDeg temp
                                                              (tanLisp brg)))))
                          )
                        )
                 ) ;end else construct brg=90 or brg=270
               )
      )
    ) ; if flag=1
    (list lat2 lng2)
  ) ;let
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(defun getRngBrg (lat1 lng1 lat2 lng2 flag)
  ; flag = 0 great circle
  ;      = 1 rhumbline
  (let* ((rng 0) (brg 0) (dlat (abs(- lat1 lat2)))
         (dlng (abs(fnCor(- lng1 lng2)))) (x) (y) (lat1t) (clat2 (cosLisp lat2 ))
        )
    (if (>= (+ dlat dlng) 0.0001)
      (progn (if (= flag 0)
                 (progn (if (= dlng 0)
                            (progn (if (< lat1 lat2)
                                       (setq brg 0)
                                       (setq brg 180)
                                   )
                                   (setq rng dlat)
                            )
                            (progn (if (= (abs lat1) 90)
                                       (setq lat1t (* 89.9
                                                      (/ lat1 (abs lat1))))
                                       (setq lat1t lat1)
                                   )
                                   (setq clat1 (cosLisp lat1t))
                                   (setq slat1 (sinLisp lat1))
                                   (setq slat2 (sinLisp lat2))
                                   (setq rng (acosLisp (fnAbs(+ (* slat1 slat2)
                                                                (* clat1 clat2
(cosLisp dlng))))))
                                   (setq brg (acosLisp (fnAbs(/ (- slat2
                                                                   (* slat1 (cosLisp


rng)))
                                                                (* clat1 (sinLisp

rng))))))
                                   (if (or (>= lng2 -90)
                                           (<= lng1 90))  ;other than east or west
dateline crossing
                                     (progn (if (< lng2 lng1)
                                                (setq brg (- 360 brg))

; is this progn necessary.
                                            (progn (if (and (> lng2 90)
                                                            (< lng1 -90))
                                                       (setq brg (- 360 brg)))
                                                   ))
                                     ))
                            ))
                 )) ;if flag=0
              (if (= flag 1)
                (progn (if (or (= (abs lat1) 90)
                               (= (abs lat2) 90)) ;North or South pole
                           (progn (setq rng dlat)
                                  (if (> lat2 lat1)
                                    (setq brg 0)
                                    (setq brg 180))
                           )
                           (progn (setq x (log (/ (tanLisp (- 45 (* 0.5 lat1)))
                                                  (tanLisp (- 45 (* 0.5 lat2))))))
                                  (setq x (* x GLB.radToDeg))
                                  (setq y (fnCor (- lng2 lng1)))
                                  (setq brg (fnArctan x y))
                                  (if (or (= brg 90)
                                          (= brg 270))
                                    (setq rng (* clat2 dlng))
                                    (setq rng (abs (/ dlat (cosLisp brg))))
                                  )
                            ))
                ))
       ))
    (setq rng (* 60.1077 rng))
    (list rng brg)
  ) ;let
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq detectionReporter (kindof generalPlatform))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (exist detectionReporter) (init-list)
  (usual-exist init-list)

  (have 'report-type       'POS+VEL  )   ;  POS-pos only, POS+VEL-pos + vel
  (have 'positional-unc     10       )   ;  nm
  (have 'speed-unc           3       )   ;  kts
  (have 'course-unc         30       )   ;  degrees

  (have 'expRepDelay  5)
  (have 'FAR          0.9)

  (have 'startTrackp  1)

  (setq  dataNameList (append dataNameList
                       '(:reports
                          (report-type positional-unc speed-unc course-unc
                           expRepDelay FAR startTrackp))))
  (setq  dataValueList (append dataValueList
                        (:reports
                          (,report-type ,positional-unc ,speed-unc ,course-unc
                            ,expRepDelay ,FAR ,startTrackp))))
  (setq  dataTypeList (append dataTypeList
                       '(:reports (data data data data data data data))))
  (setq  dataTextList (append dataTextList
                       '(:reports ("REPORT TYPE" "POSITION ERROR" "SPEED ERROR"
                                   "COURSE ERROR" "Exp Processing Delay"
                                   "False Alarm Rate" "Track Initiation"))))
  (setq  dataTemplateText (append dataTemplateText
                           '(:reports
                              (("POS or POS+VEL" "[POS , POS+VEL]")
                               ("Nautical Miles" "[0 - 1000]")
                               ("Knots"          "[0 - 30]")
                               ("Degrees"        "[0 - 180]")
                               ("Hours"          "[0 - 100]")
                               ("FA per Day"     "[0 - unlimited]")
                               ("Single Contact to Track"     "0 - no, 1 -
yes")))))
  (let* ((rb-designators  '(reports-rb ))
         (rb-values        (list dummyRBdata))
         (rb-names        '(dummyRBdata))
        )
        (update-rb-lists rb-designators rb-values rb-names)
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setForStart detectionReporter) ()
 (usual-setForStart)
 (have 'computedFAR FAR)
```

```
)

;;;
```

```
                                        :receipt-time       time
                                        :track-association targetName
                                        :sensor            sensor-field
                                        :categorization    cat
                                        :spatial-data      spatial-data-field
                                        :altitude          hgt
                                        :HFDFp             HFDFp
                                      )
          )
    )
)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;

(defobfun (make-false-obu-report detectionReporter)(lat lng uncertainty ctime
targetType)
  (let ((falseTargetState (getFalsetargetState lat lng uncertainty targetType)))
       (make-obu-report ctime falseTargetState)
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getFalsetargetState detectionReporter) (lat lng uncertainty targetType)
  (multiple-value-bind
        (lat lng)
        (contactPositionDraw lat lng uncertainty)
        (return-from getFalseTargetState
            (make-platformStateParms
               :lat lat
               :lng lng
               :hgt 0
               :cse (* (random 1.0) 360)
               :spd 10
               :platformName 'FA
               :targetType targetType
            )
        )
  )
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct coast-event
    time
    object
    procedure
    data
    updateList
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct coast-message
    send-time                   ; Time message sent.
    receipt-time                ; Time message received at last node.
    type                        ; E.g. 'Detect-msg
    sender
    content
    size                        ; MBs
    transmission-path
    transmission-count
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct obu-contact
    id
    receipt-time
    track-association
    sensor
    categorization
    spatial-data
    altitude
    HFDFp
)

(defstruct obu-track
    id
    number-of-contacts
    head-state-covariance
    head-contact-id
    head-spatial-data
    tail-state-covariance
    tail-contact-id
    tail-spatial-data
    altitude
    (contacts nil :type list)
)

(defstruct obu-data-field
    type
    obs-time
    lat
    lng
    brg
    maj

    min
    cse
    cse-unc
    spd
    spd-unc
)

(defstruct obu-state-field
    ref-lat
    ref-lng
    (state      nil :type (array float    4  ))
    (covariance nil :type (array float '(4 4)))
)

(defstruct obu-sensor
    name
    single-report-to-trackp
)

(defstruct obu-category
    pinnedp
    lockedp
    assoc-count
    cluster-count
    ambiguity-list
    ambiguity-resolution
    ambiguity-resolution-time
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(defun contactPositionDraw (lat lng rng)
   ; a draw from a circular normal -- rng is interpreted as a 2 sigma confidence
   ; interval
   (let ((randomRange (min (* (/ rng 2) (sqrt (* -2 (log (random 1.0)))))
                           (* 4 rng)))
         (randomAngle (randomCse))
        )
        (return-from contactPositionDraw
                     (values-list
                        (getLatLng lat lng randomRange randomAngle
*greatCircle*)))
   )

)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;  SPA FUNCTIONS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun expandAOUpos (vel tau timeLate)
  (let ((ratio  (/ timeLate tau))
        (factor (* 2 pi vel vel tau tau)))
    (* factor (+ ratio -1 (exp (* -1 ratio))))
  )
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun expandAOUposVel (vel tau timeLate)
  (let* ((ratio   (/ timeLate tau))
         (factor1 (* 2 pi vel vel tau tau))
         (factor2 (* 2   (exp (* -1 ratio))))
         (factor3 (* 0.5 (exp (* -2 ratio))))
        )
    (* factor1 (+ ratio factor2 -1.5 (* -1 factor3)))
  )
)

;;;;;;;;;;;;;;;;;;;
```

APPENDIX C: ITERATIONS TO ACHIEVE STEADY STATE GAIN VALUES

The model in this appendix calculates the percent differece between succesive iterated values of Kalman gain K
These values are tabulated for various ranges of interarrival time and sensor AOU size.

$n := 0..25 \qquad \Delta := 50 \qquad u := 100 \qquad \alpha := .25 \qquad \sigma := \sqrt{50} \qquad r3 := \frac{\sigma^2}{2\cdot\alpha} \qquad r1 := .25\cdot u^2$

$v1_0 := r1 \qquad v2_0 := 0 \qquad v3_0 := r3 \qquad \phi 2 := \frac{1}{\alpha}\cdot[1 - \exp[-\alpha\cdot\Delta]] \qquad \phi 3 := \exp[-\alpha\cdot\Delta]$

$$q1 := \frac{\sigma^2}{\alpha^2}\cdot\left[\Delta - \left[\frac{1}{\alpha}\cdot[[2\cdot[1 - \exp[-\alpha\cdot\Delta]] - .5\cdot[1 - \exp[-\alpha\cdot 2\cdot\Delta]]]\right]\right] \qquad q2 := \frac{\sigma^2}{\alpha^2}\cdot[.5 - \exp[-\alpha\cdot\Delta] + [.5\cdot\exp[-\alpha\cdot 2\cdot\Delta]]] \qquad q3 := .5\cdot\frac{\sigma^2}{\alpha}\cdot[1 - \exp[-\alpha\cdot 2\cdot\Delta]$$

$$\begin{bmatrix} v1_{n+1} \\ v2_{n+1} \\ v3_{n+1} \end{bmatrix} = \begin{bmatrix} r1\cdot\dfrac{\left[\left[v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1\right]\cdot\left[\phi 3^2\cdot v3_n + q3\right] + \left[v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1\right]\cdot r3 - \left[\left[v2_n + \phi 2\cdot v3_n\right]\cdot\phi 3 + q2\right]^2\right]}{\left[v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1 + r1\right]\cdot\left[\phi 3^2\cdot v3_n + q3 + r3\right] - \left[\left[v2_n + \phi 2\cdot v3_n\right]\cdot\phi 3 + q2\right]^2} \\ \left[\left[v2_n + \phi 2\cdot v3_n\right]\cdot\phi 3 + q2\right]\cdot r1\cdot\dfrac{r3}{\left[v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1 + r1\right]\cdot\left[\phi 3^2\cdot v3_n + q3 + r3\right] - \left[\left[v2_n + \phi 2\cdot v3_n\right]\cdot\phi 3 + q2\right]^2} \\ r3\cdot\dfrac{\left[-\left[\left[v2_n + \phi 2\cdot v3_n\right]\cdot\phi 3 + q2\right]^2 + \left[v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1\right]\cdot\left[\phi 3^2\cdot v3_n + q3\right] + r1\cdot\left[\phi 3^2\cdot v3_n + q3\right]\right]}{\left[v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1 + r1\right]\cdot\left[\phi 3^2\cdot v3_n + q3 + r3\right] - \left[\left[v2_n + \phi 2\cdot v3_n\right]\cdot\phi 3 + q2\right]^2} \end{bmatrix}$$

$$kl_n := \frac{v1_n}{r1} \qquad t := 1..5 \qquad Diff_t := \frac{kl_t - kl_{t-1}}{kl_t}\cdot 100$$

Sensor AOU = 1nm

$\Delta := 1\cdot 10^{-3}$

$Diff_t$
-99.980002
-49.955032
-33.266773
-24.912739
-19.892447
-16.539079
-14.138292
-12.332923
-10.924554
-9.794141
-8.865931
-8.089429
-7.429679
6.861717
-6.367239
-5.932517
-5.54706
-5.202706
-4.893016
-4.612839
-4.358007
-4.12511
-3.911334
-3.714333
-3.532138

$\Delta := .01$

$Diff_t$
-98.037619
-45.804206
-27.592024
-18.198738
-12.55896
-8.932986
-6.525805
-4.903215
-3.800901
-3.04773
-2.529217
-2.167554
-1.909555
1.718907
-1.57099
-1.449334
-1.343167
1.245689
-1.152855
-1.062508
0.973753
-0.886509
-0.801183
-0.718437
-0.639022

$\Delta := .05$

$Diff_t$
-66.575636
-12.535059
-6.501015
-6.359547
-6.09602
-5.155953
-3.855277
-2.560588
-1.510314
-0.786681
-0.356929
-0.137247
-0.042686
0.010612
-0.003816
-0.004216
-0.005
-0.004674
-0.003587
-0.002352

$\Delta := .1$

$Diff_t$
-33.154823
-7.528865
-9.446393
-7.051605
-3.604176
-1.284654
-0.298693
-0.035621
-0.004605
-0.008734

$\Delta := .5$

$Diff_t$
-1.894728
-4.579219
-0.002639
-0.008118
$-5.581487\cdot 10^{-6}$

$\Delta := 1$

$Diff_t$
-0.469314
-0.821639
-0.031427
$-8.463799\cdot 10^{-4}$
$-2.082199\cdot 10^{-5}$

Sensor AOU = 10nm

Δ =0.01	Δ =0.05	Δ =0.1	Δ =0.5	Δ =1	Δ =5
Diff$_i$	Diff$_i$	Diff$_i$	Diff$_i$	Diff$_i$	Diff$_i$
-99.979987	-99.500457	-98.023693	-65.885691	-32.043382	-1.873284
-49.955028	-48.894589	-45.802954	-12.837194	-6.917853	-0.749395
-33.266776	-31.73098	-27.595158	-5.388178	-3.044708	-0.013501
-24.912704	-22.951201	-18.165175	-2.947916	-0.490318	$-1.586052 \cdot 10^{-4}$
-19.892292	-17.546569	-12.43854	-1.356479	-0.027738	$-1.635119 \cdot 10^{-6}$
-16.53869	-13.849295	-8.68271	-0.483994		
-14.137527	-11.145522	-6.118373	-0.132892		
-12.331612	-9.078576	-4.328909	-0.028186		
-10.922505	-7.450049	-3.065327	-0.005372		
-9.791141	-6.140237	-2.16778	-0.001726		
-8.861754	-5.072017	-1.528851			
-8.083836	-4.192852	-1.074192			
-7.422426	-3.465175	-0.751363			
-6.852556	-2.860974	-0.522938			
-6.355922	-2.358609	-0.362028			
-5.918804	-1.94088	-0.24926			
-5.530716	-1.593814	-0.170673			
-5.18351	-1.30588	-0.116232			
-4.870759	-1.067456	-0.078746			
-4.58733	-0.870461	-0.053094			
-4.329072	-0.708084				
-4.092597	-0.574577				
-3.875111	-0.465089				
-3.674291	-0.375535				
-3.488188	-0.302477				

Sensor AOU = 100nm

Δ =0.1	Δ =0.5	Δ =1	Δ =5	Δ =10	Δ =50
Diff$_i$	Diff$_i$	Diff$_i$	Diff$_i$	Diff$_i$	Diff$_i$
-99.979843	-99.484887	-97.923273	-65.624462	-36.291752	-6.493503
-49.955004	-48.896967	-45.855825	-14.77194	-4.450593	-0.141054
-33.266816	-31.738027	-27.649854	-3.130706	-0.269804	$-2.531692 \cdot 10^{-4}$
-24.912377	-22.923018	-18.018772	-0.559593	-0.012582	$-4.543601 \cdot 10^{-7}$
-19.89094	-17.451178	-12.047199	-0.091712	$-5.467805 \cdot 10^{-4}$	$-8.154349 \cdot 10^{-10}$
-16.535544	-13.674001	-8.113983			
-14.131812	-10.895156	-5.470752			
-12.322629	-8.769084	-3.685715			
-10.909682	-7.102159	-2.480428			
-9.774059	-5.774855	-1.667759			
-8.840153	-4.70748	-1.120598			
-8.057607	-3.843648	-0.75261			
-7.391589	-3.141649	-0.505316			
-6.817245	-2.569625	-0.339214			
-6.31636	-2.102692	-0.227685			
-5.875281	-1.721104	-0.152814			
-5.483573	-1.409027	-0.102559			
-5.13312	-1.153674	-0.068829			
-4.817516	-0.944669	-0.046192			
-4.531635	-0.773565	-0.030999			
-4.271329	-0.63347				
-4.033201	-0.518756				
-3.814445	-0.424819				
-3.61272	-0.347893				
-3.42606	-0.284898				

APPENDIX D: KALMAN GAIN VERSUS INTERARRIVAL TIME

This appendix shows the relationship between the Kalman gain and the interarrival time. This information can be used to select a range of time values to tabulate preprocessed variance data.

$n = 0..25 \qquad \Delta = 10 \qquad u = 50 \qquad \alpha = .25 \qquad \sigma = \sqrt{50}$

$r1 = .25 \cdot u^2 \qquad r3 = \frac{\sigma^2}{2 \cdot \alpha}$

$v1_0 = r1 \qquad v2_0 = 0 \qquad v3_0 = r3 \qquad \phi 2 = \frac{1}{\alpha} \cdot [1 - \exp[-\alpha \cdot \Delta]] \qquad \phi 3 = \exp[-\alpha \cdot \Delta]$

$q3 = .5 \cdot \frac{\sigma^2}{\alpha} \cdot [1 - \exp[-\alpha \cdot 2 \cdot \Delta]]$

$q1 = \frac{\sigma^2}{\alpha^2} \cdot \left[\Delta - \left[\frac{1}{\alpha} \cdot [[2 \cdot [1 - \exp[-\alpha \cdot \Delta]] - .5 \cdot [1 - \exp[-\alpha \cdot 2 \cdot \Delta]]]]\right]\right] \qquad q2 = \frac{\sigma^2}{\alpha^2} \cdot [.5 - \exp[-\alpha \cdot \Delta] + .5 \cdot \exp[-\alpha \cdot 2 \cdot \Delta]]$

$$\begin{bmatrix} v1_{n+1} \\ v2_{n+1} \\ v3_{n+1} \end{bmatrix} = \begin{bmatrix} r1 \cdot \frac{\left[\left[v1_n + 2 \cdot \phi 2 \cdot v2_n + \phi 2^2 \cdot v3_n + q1\right] \cdot \left[\phi 3^2 \cdot v3_n + q3\right] + \left[v1_n + 2 \cdot \phi 2 \cdot v2_n + \phi 2^2 \cdot v3_n + q1\right] \cdot r3 - \left[\left[v2_n + \phi 2 \cdot v3_n\right] \cdot \phi 3 + q2\right]^2\right]}{\left[v1_n + 2 \cdot \phi 2 \cdot v2_n + \phi 2^2 \cdot v3_n + q1 + r1\right] \cdot \left[\phi 3^2 \cdot v3_n + q3 + r3\right] - \left[\left[v2_n + \phi 2 \cdot v3_n\right] \cdot \phi 3 + q2\right]^2} \\ \left[\left[v2_n + \phi 2 \cdot v3_n\right] \cdot \phi 3 + q2\right] \cdot r1 \cdot \frac{r3}{\left[v1_n + 2 \cdot \phi 2 \cdot v2_n + \phi 2^2 \cdot v3_n + q1 + r1\right] \cdot \left[\phi 3^2 \cdot v3_n + q3 + r3\right] - \left[\left[v2_n + \phi 2 \cdot v3_n\right] \cdot \phi 3 + q2\right]^2} \\ r3 \cdot \frac{\left[-\left[\left[v2_n + \phi 2 \cdot v3_n\right] \cdot \phi 3 + q2\right]^2 + \left[v1_n + 2 \cdot \phi 2 \cdot v2_n + \phi 2^2 \cdot v3_n + q1\right] \cdot \left[\phi 3^2 \cdot v3_n + q3\right] + r1 \cdot \left[\phi 3^2 \cdot v3_n + q3\right]\right]}{\left[v1_n + 2 \cdot \phi 2 \cdot v2_n + \phi 2^2 \cdot v3_n + q1 + r1\right] \cdot \left[\phi 3^2 \cdot v3_n + q3 + r3\right] - \left[\left[v2_n + \phi 2 \cdot v3_n\right] \cdot \phi 3 + q2\right]^2} \end{bmatrix}$$

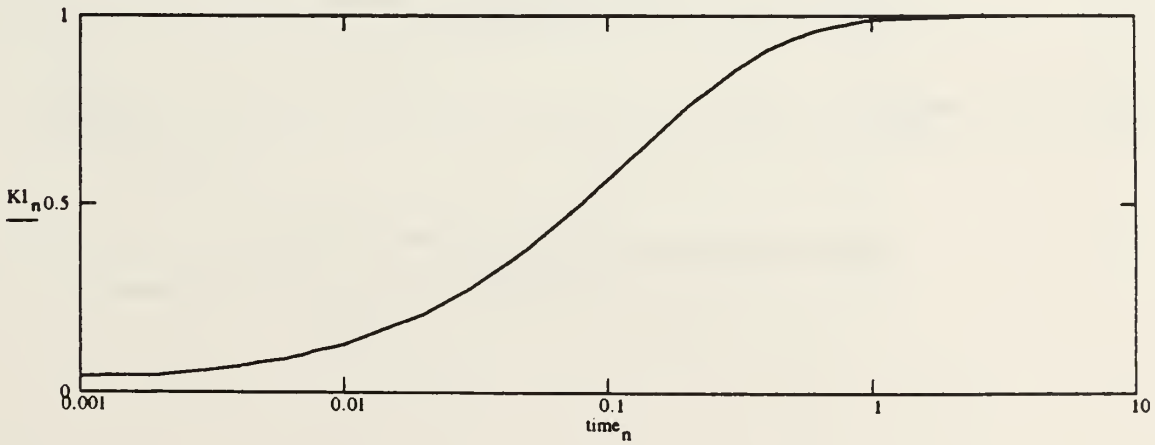
$k1_n = \frac{v1_n}{r1} \qquad k2a_n = \frac{v2_n}{r3} \qquad k2b_n = \frac{v2_n}{r1} \qquad k3_n = \frac{v3_n}{r3}$

APPENDPRN(Kalman) = [Δ u $k1_{25}$ $k2a_{25}$ $k2b_{25}$ $k3_{25}$]

KDATA = READPRN(Kalman)

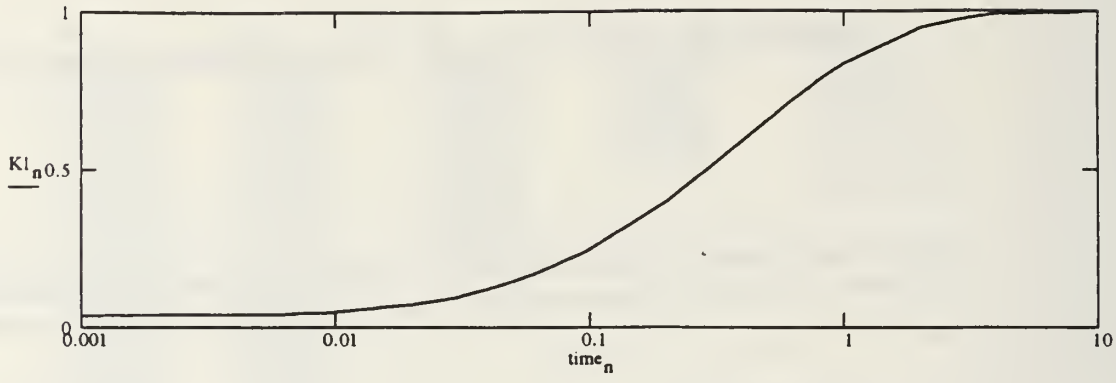
time = KDATA⟨0⟩ K1 = KDATA⟨2⟩ n = 0..last(KDATA⟨0⟩)

Sensor AOU = 1nm



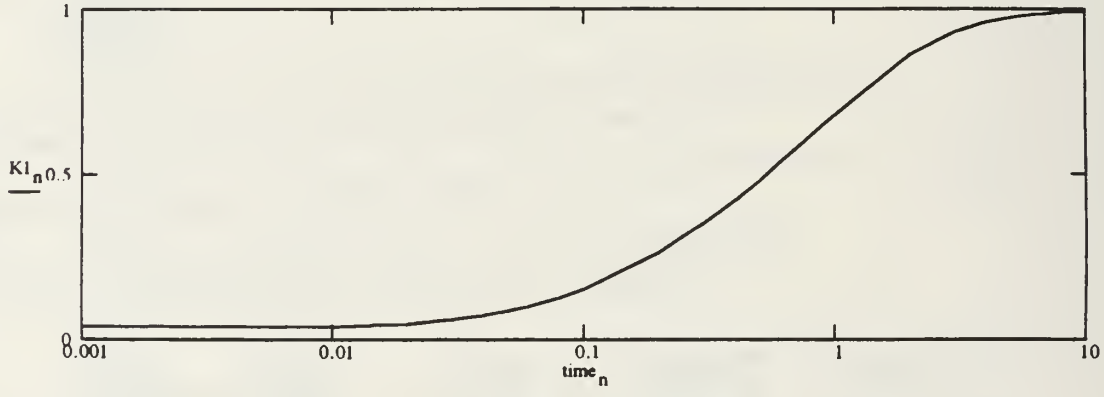
KDATA := READPRN(Kalman5) time := KDATA^<0> KI := KDATA^<2> n := 0..last(KDATA^<0>)

Sensor AOU =5nm



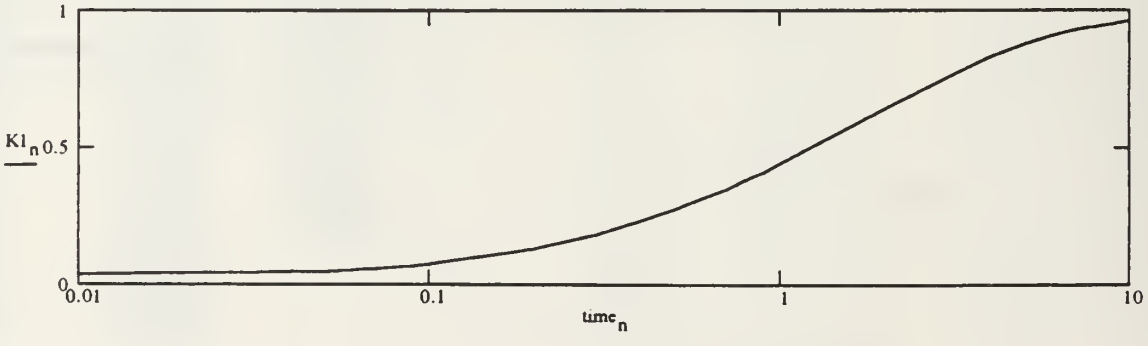
KDATA := READPRN(Kalman10) time := KDATA^<0> KI := KDATA^<2> n := 0..last(KDATA^<0>)

Sensor AOU =10nm



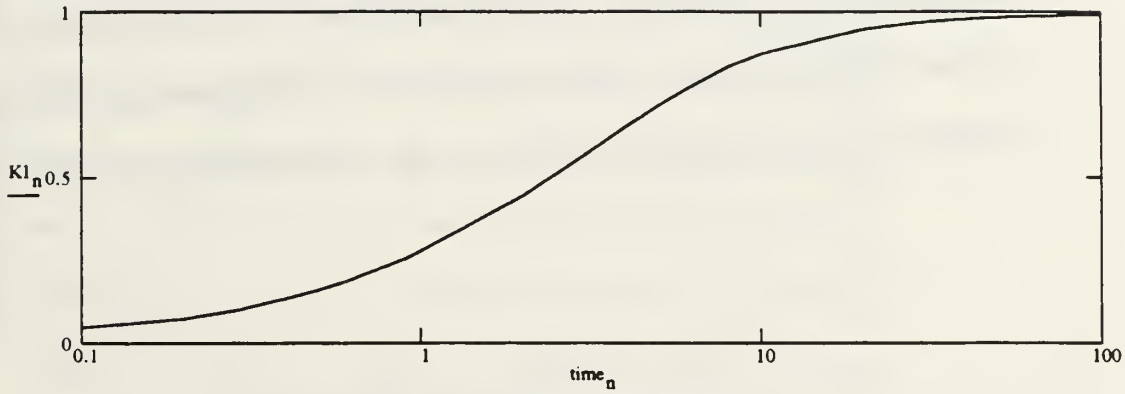
KDATA := READPRN(Kalman25) time := KDATA^<0> KI := KDATA^<2> n := 0..last(KDATA^<0>)

Sensor AOU =25nm



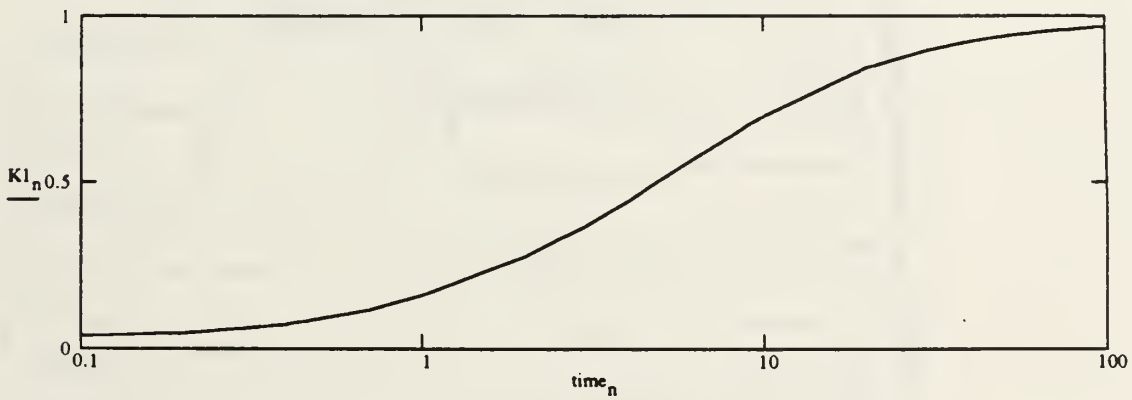
KDATA := READPRN(Kalman50) time := KDATA<0> Kl := KDATA<2> n := 0..last(KDATA<0>)

Sensor AOU =50nm



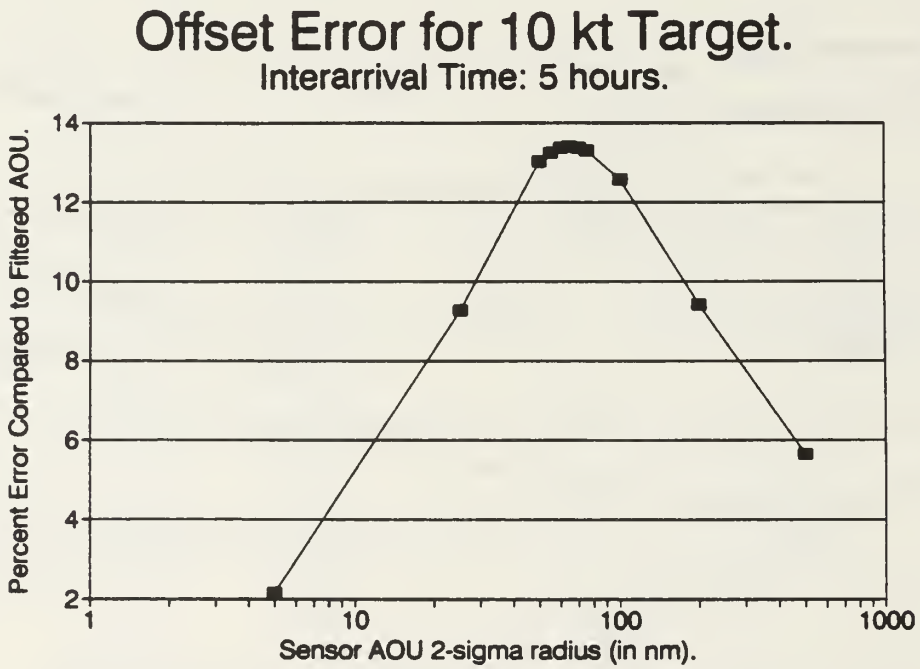
KDATA := READPRN(Kalm100) time := KDATA<0> Kl := KDATA<2> n := 0..last(KDATA<0>)

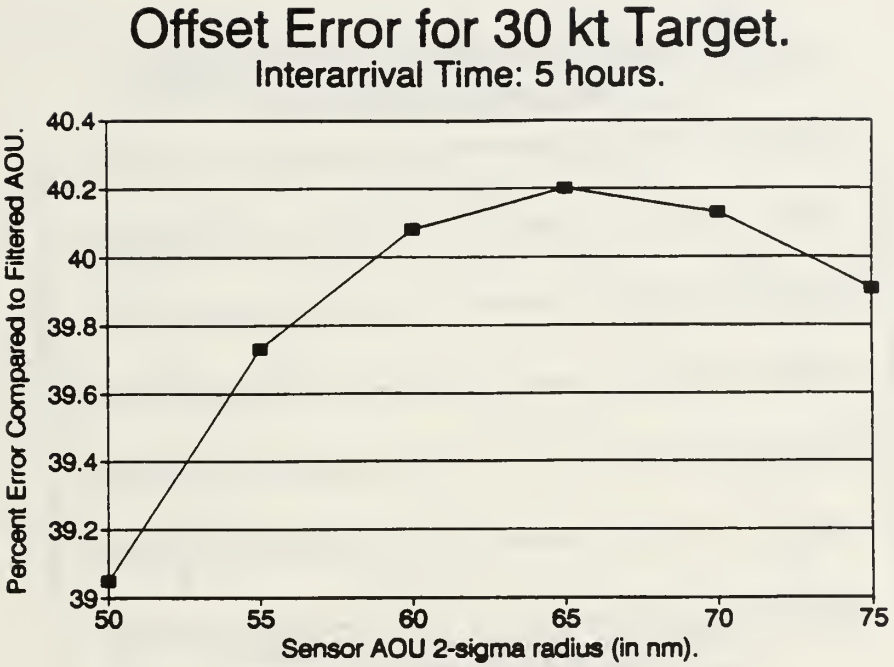
Sensor AOU =100nm

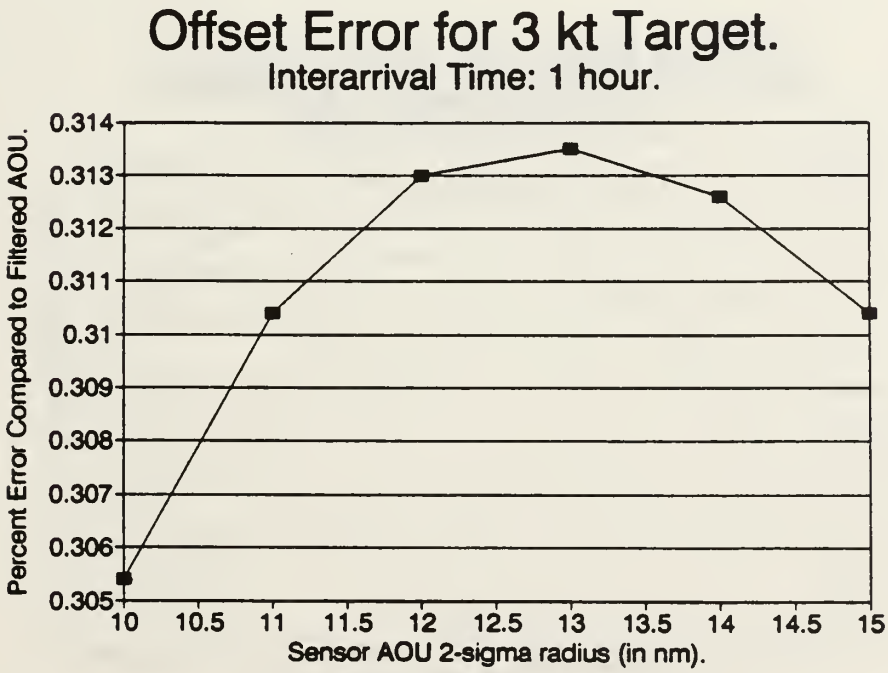


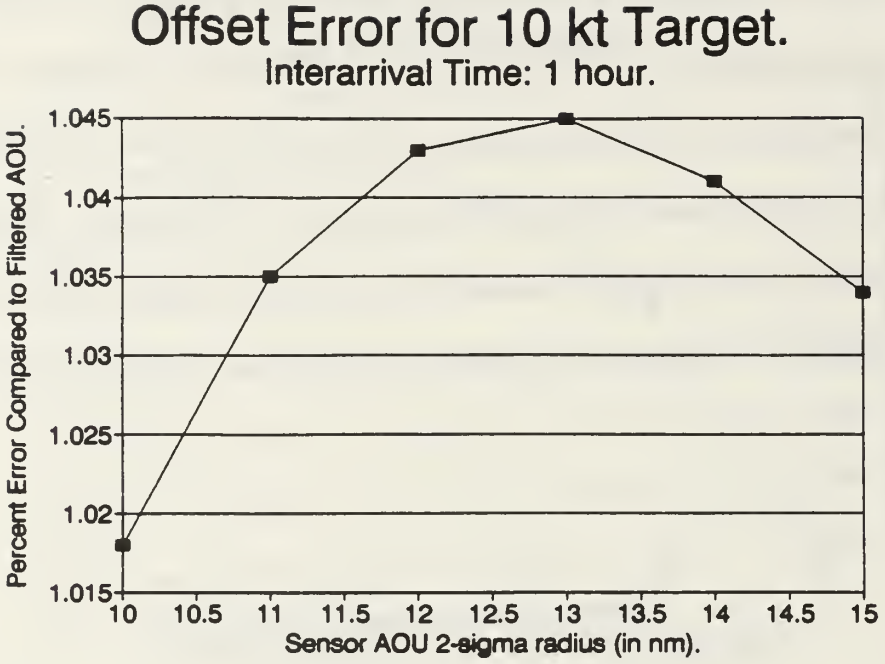
APPENDIX E: OFFSET ERROR OF FILTER POSITION DISTRIBUTION

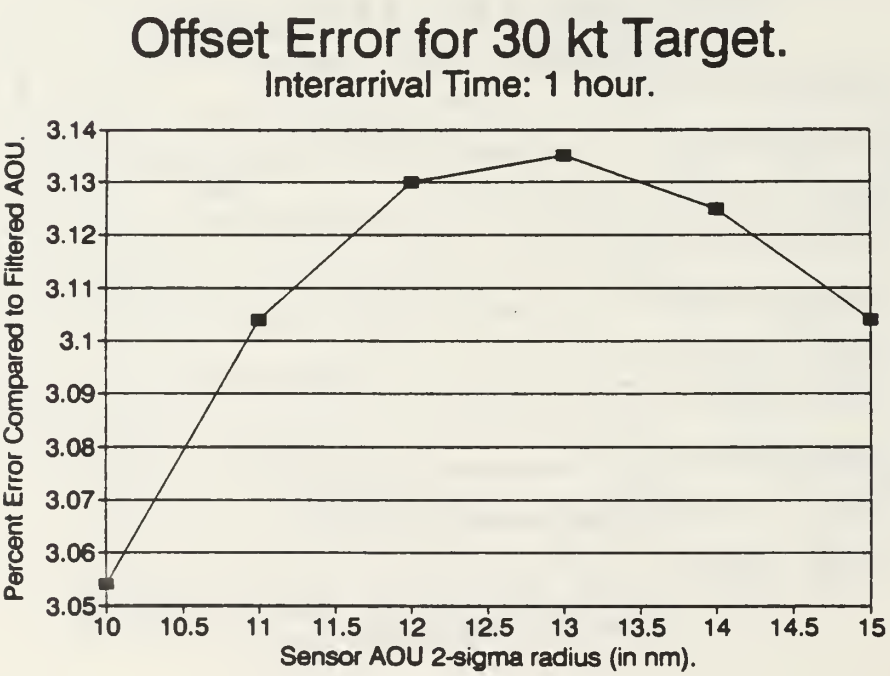
This appendix provides indications of how far from the target's true position the filtered position is displaced by the IOU process adjustment of velocity to account for the possibility of a manuever. This assumes the target is actually moving with constant course and speed. The first graph cover's the entire range from 1nm to 1000nm while succeding graphs show only the neighborhood of the maximum value

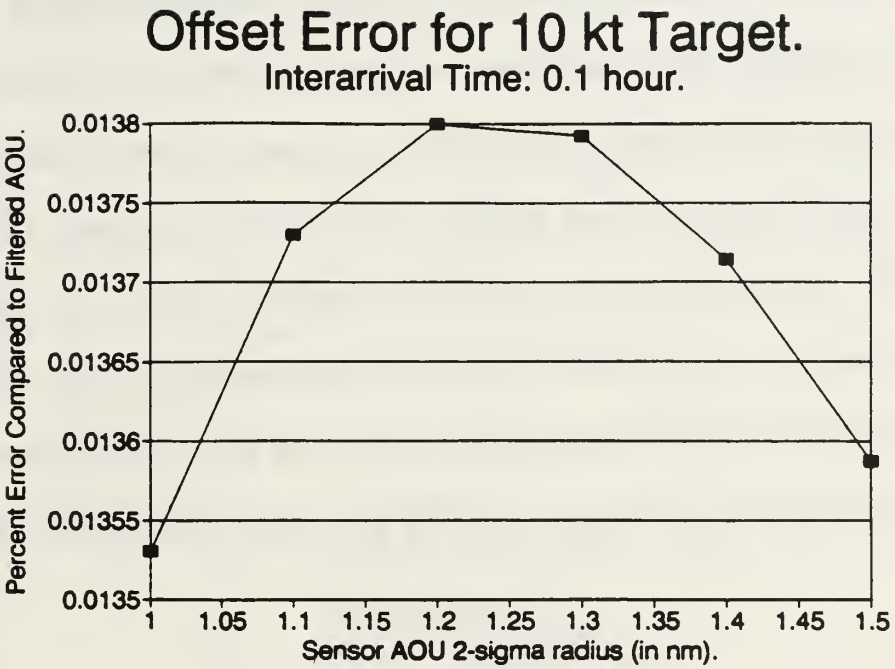


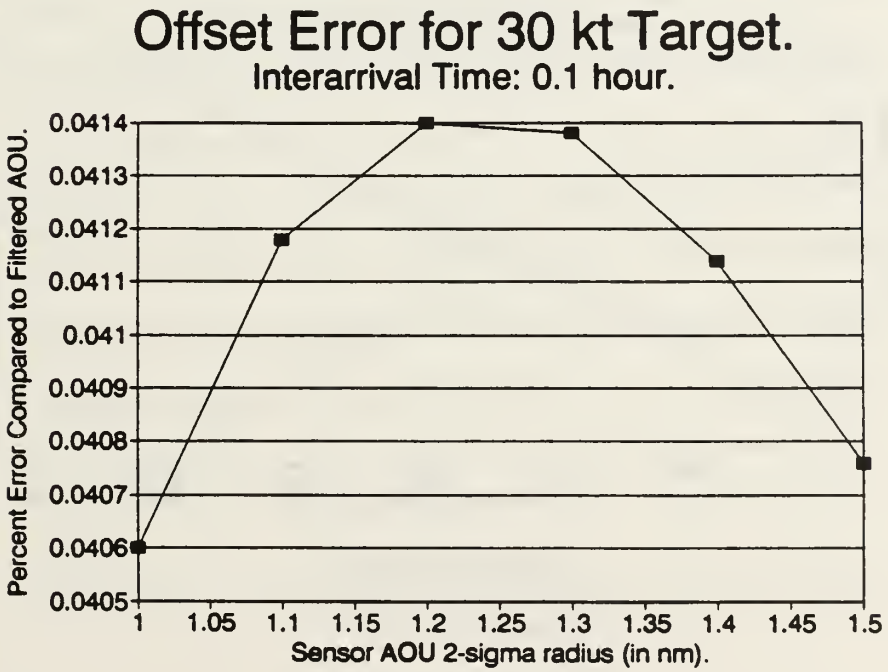

Offset Error for 30 kt Target.
Interarrival Time: 5 hours.
Percent Error Compared to Filtered AOU.
40.4
40.2
40
39.8
39.6
39.4
39.2
39
50
55
60
65
70
75
Sensor AOU 2-sigma radius (in nm).


Offset Error for 3 kt Target.
Interarrival Time: 1 hour.
Percent Error Compared to Filtered AOU.
0.314
0.313
0.312
0.311
0.31
0.309
0.308
0.307
0.306
0.305
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15
Sensor AOU 2-sigma radius (in nm).


Offset Error for 10 kt Target.
Interarrival Time: 1 hour.
Percent Error Compared to Filtered AOU.
1.045
1.04
1.035
1.03
1.025
1.02
1.015
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15
Sensor AOU 2-sigma radius (in nm).

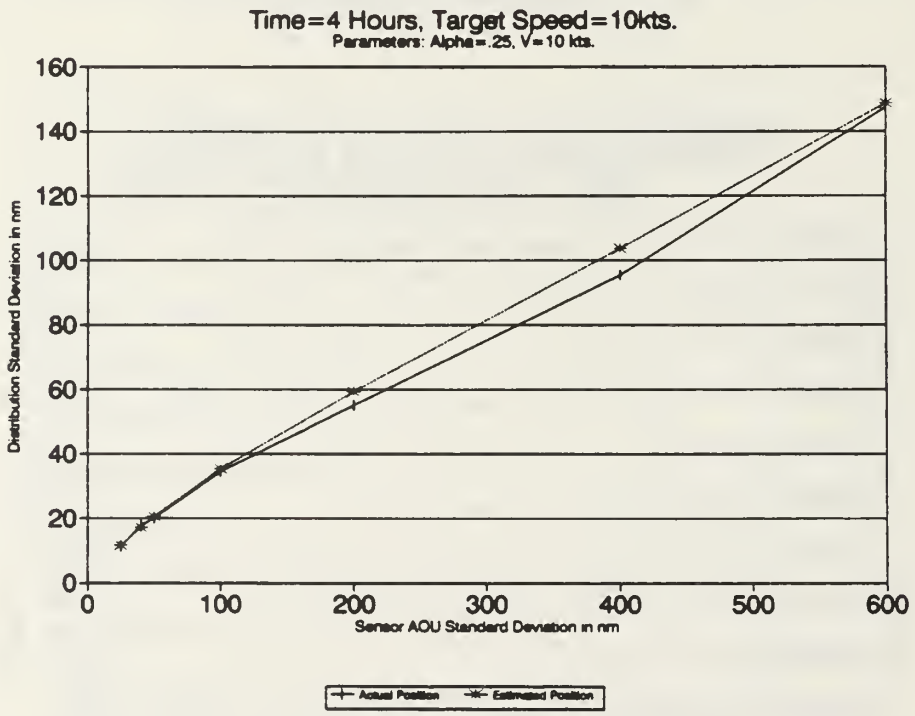

Offset Error for 30 kt Target.
Interarrival Time: 1 hour.
Percent Error Compared to Filtered AOU.
3.14
3.13
3.12
3.11
3.1
3.09
3.08
3.07
3.06
3.05
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15
Sensor AOU 2-sigma radius (in nm).

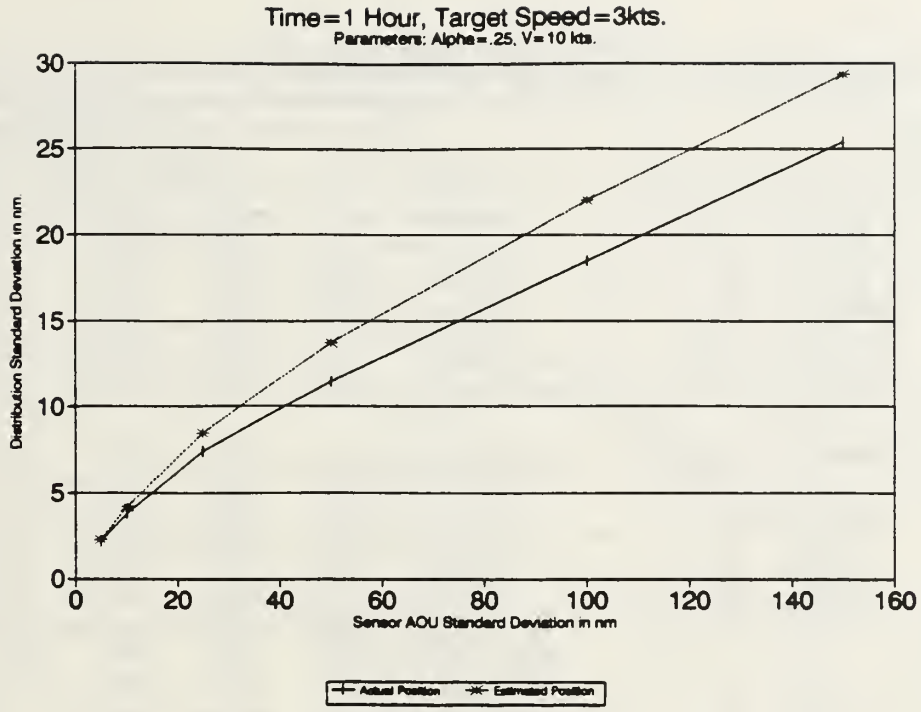

Offset Error for 10 kt Target.
Interarrival Time: 0.1 hour.
Percent Error Compared to Filtered AOU.
0.0138
0.01375
0.0137
0.01365
0.0136
0.01355
0.0135
1
1.05
1.1
1.15
1.2
1.25
1.3
1.35
1.4
1.45
1.5
Sensor AOU 2-sigma radius (in nm).

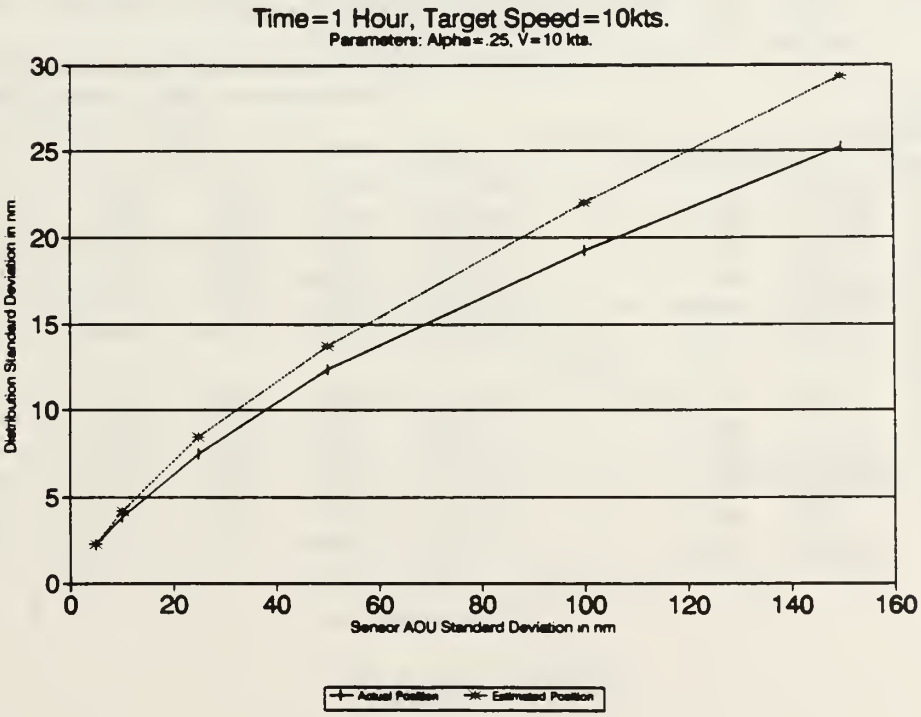

Offset Error for 30 kt Target.
Interarrival Time: 0.1 hour.
Percent Error Compared to Filtered AOU.
0.0414
0.0413
0.0412
0.0411
0.041
0.0409
0.0408
0.0407
0.0406
0.0405
1
1.05
1.1
1.15
1.2
1.25
1.3
1.35
1.4
1.45
1.5
Sensor AOU 2-sigma radius (in nm).

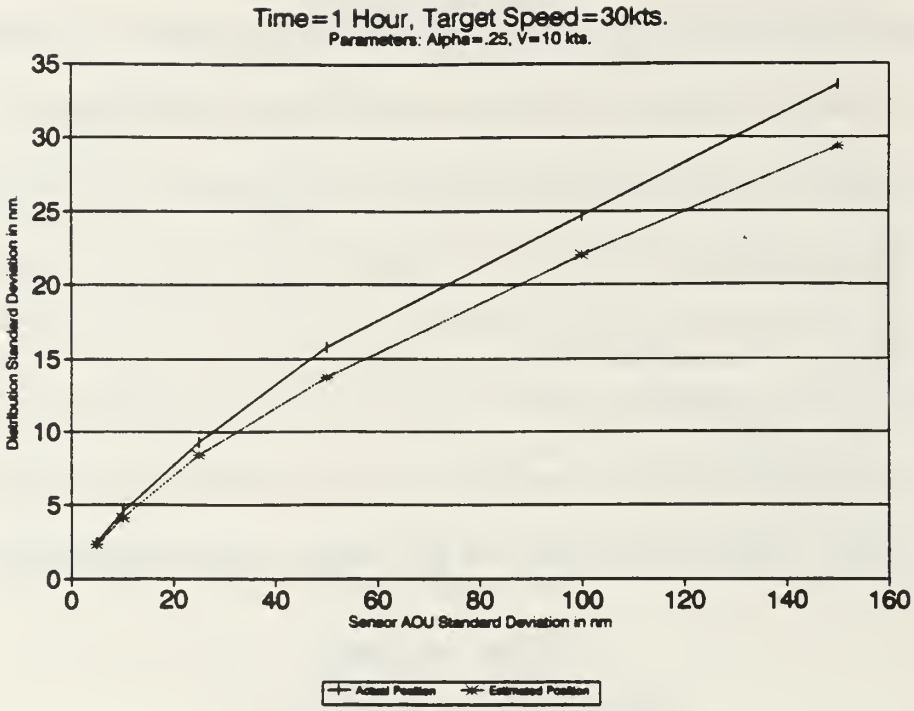
APPENDIX F: ERROR BETWEEN FILTERED AND ESTIMATED SPA SIZE

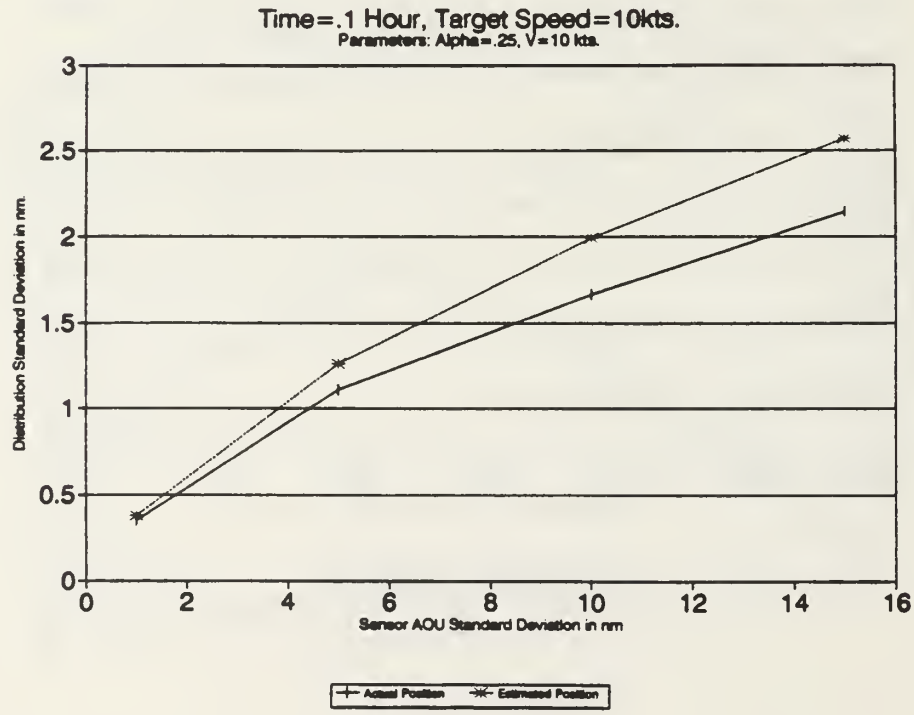
This appendix shows the difference in standard deviation between the actual filtered position distribution and the estimated distribution. The data tabulated shows more complete data for the cases modeled. This includes the observed AOU (Obs.AOU) and the predicted AOU (PRED AO); the mean value of the standard deviation calculated for 30 runs of 48 hours of 300 observations for the position predicted by the IOU model, the contacts position, and the filtered position; as well as the actual mean speed of the contact over the 30 runs and the mean filtered value.




Time=1 Hour, Target Speed=3kts.
Parameters: Alpha=.25, V=10 kts.
Distribution Standard Deviation in nm
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
Sensor AOU Standard Deviation in nm
Actual Position
Estimated Position


Time=1 Hour, Target Speed=10kts.
Parameters: Alpha=.25, V=10 kts.
Distribution Standard Deviation in nm
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
Sensor AOU Standard Deviation in nm
Actual Position
Estimated Position


Time=1 Hour, Target Speed=30kts.
Parameters: Alpha=.25, V=10 kts.
Distribution Standard Deviation in nm
0
5
10
15
20
25
30
35
0
20
40
60
80
100
120
140
160
Sensor AOU Standard Deviation in nm
Actual Position
Estimated Position


Time=.1 Hour, Target Speed=10kts.
Parameters: Alpha=.25, V=10 kts.
Distribution Standard Deviation in nm
0
0.5
1
1.5
2
2.5
3
0
2
4
6
8
10
12
14
16
Sensor AOU Standard Deviation in nm
Actual Position
Estimated Position

Data for on IOU process motion model beta=.25 and sigma based on 10 knot average speed

Time interval between observations of 4 hours:

Standard Deviation of error in nautical miles:

Prediction mean	Contact mean	Filter mean	SPA	%Error	Obs. AOU	Pred. AO	Speed in knots. Contact mean	Filtered mean
14.941	11.957	11.032	11.499	-4.06122	25	30.684	2.959	3.044
20.041	19.071	16.716	17.18	-2.70081	40	34.216	2.975	3.145
22.194	23.148	19.693	20.594	-4.37506	50	36.52	3.078	3.078
33.145	43.599	32.113	35.113	-8.54384	100	47.234	3.02	2.635
52.749	81.301	53.04	59.109	-10.2675	200	67.285	2.92	2.118
92.721	160.022	93.447	103.544	-9.75141	400	108.788	2.992	1.881
149.547	245.976	151.585	148.352	2.179276	600	152.841	3.061	1.895
21.938	11.925	11.197	11.499	-2.62632	25	30.684	10.133	7.482
27.577	19.357	17.561	17.18	2.217695	40	34.216	10.026	7.087
27.27	22.54	19.703	20.594	-4.3265	50	36.52	9.997	6.81
40.463	43.917	34.288	35.113	-2.34956	100	47.234	9.952	6.369
57.402	84.078	54.921	59.109	-7.08522	200	67.285	9.943	5.999
95.207	154.341	95.083	103.544	-8.17141	400	108.788	10.307	5.925
146.205	243.202	147.258	148.352	-0.73744	600	152.841	10.076	5.958
56.41	13.4	13.682	11.499	18.98426	25	30.684	29.933	21.703
59.391	21.688	22.186	17.18	29.13853	40	34.216	29.931	20.036
63.26	26.691	26.537	20.594	28.85792	50	36.52	29.962	20.124
78.533	47.705	47.533	35.113	35.37151	100	47.234	30.019	18.783
101.683	89.644	78.28	59.109	32.4333	200	67.285	30.001	18.033
133.696	169.918	121.254	103.544	17.10384	400	108.788	29.908	17.635
169.943	233.559	160.386	148.352	8.111788	600	152.841	30.072	17.556

Time interval between observations of 2 hours:

Standard Deviation of error in nautical miles:

Prediction mean	Contact mean	Filter mean	SPA	%Error	Obs. AOU	Pred. AO	Speed in knots. Contact mean	Filtered mean
7.08	4.859	4.547	4.658	-2.383	10	14.219	3.041	3.374
10.401	9.202	8.001	8.471	-5.54834	20	17.145	3.017	3.416
16.528	20.859	15.156	17.217	-11.9707	50	24.218	3.023	2.904
25.218	40.585	24.689	28.317	-12.8121	100	33.625	3.068	2.421
43.418	78.444	43.725	47.058	-7.08275	200	50.636	3.065	2.144
58.566	115.538	58.707	64.879	-9.51309	300	67.655	2.897	1.979
9.704	4.805	4.583	4.658	-1.61013	10	14.219	10.066	8.383
12.211	8.83	7.806	8.471	-7.85031	20	17.145	9.972	7.805
19.499	21.68	16.535	17.217	-3.9612	50	24.218	9.964	7.004
28.038	39.535	25.836	28.317	-8.76152	100	33.625	10.025	6.615
43.761	78.011	42.761	47.058	-9.13128	200	50.636	9.958	6.384
64.397	115.971	63.61	64.879	-1.95595	300	67.655	9.975	6.213

22.67	5.196	5.242	4.658	12.53757	10	14.219	30.061	24.285
25.584	10.066	9.858	8.471	16.37351	20	17.145	29.933	21.955
32.774	22.442	20.253	17.217	17.63373	50	24.218	30.047	20.111
47.178	43.851	35.221	28.317	24.38111	100	33.625	29.957	19.565
64.995	83.895	56.581	47.058	20.23673	200	50.636	29.938	18.814
77.334	119.128	71.016	64.879	9.459147	300	67.655	29.979	18.895

Time interval between observations of 1 hours:

Standard Deviation of error in nautical miles:

Speed in knots.

Prediction mean	Contact mean	Filter mean	SPA	%Error	Obs. AOU	Pred. AO	Contact mean	Filtered mean
3.607	2.323	2.146	2.296	-6.5331	5	6.429	3.079	3.489
5.215	4.445	3.803	4.159	-8.55975	10	8.15	2.989	3.515
8.437	10.543	7.439	8.435	-11.8079	25	12	2.96	3.119
11.984	19.878	11.43	13.716	-16.6667	50	16.684	3.032	2.737
18.691	37.993	18.477	22.018	-16.0823	100	24.227	2.984	2.329
25.46	58.34	25.401	29.352	-13.4608	150	31.113	2.991	2.215
4.568	2.393	2.243	2.296	-2.30836	5	6.429	9.988	8.922
6.018	4.461	3.846	4.159	-7.52585	10	8.15	9.904	8.387
9.107	10.411	7.472	8.435	-11.4167	25	12	10.002	7.7
13.545	19.994	12.392	13.716	-9.65296	50	16.684	10.005	7.255
19.93	37.793	19.213	22.018	-12.7396	100	24.227	10.091	7.009
25.759	54.988	25.225	29.352	-14.0604	150	31.113	9.946	6.891
9.34	2.448	2.452	2.296	6.794425	5	6.429	30.02	26.103
10.781	4.784	4.551	4.159	9.425343	10	8.15	30.055	24.381
14.764	10.961	9.315	8.435	10.43272	25	12	29.998	22.467
20.459	21.364	15.801	13.716	15.20122	50	16.684	29.948	21.356
28.395	39.358	24.728	22.018	12.30811	100	24.227	29.953	20.68
36.583	57.941	33.572	29.352	14.37721	150	31.113	30.002	20.393

Time interval between observations of .5 hours:

Standard Deviation of error in nautical miles:

Speed in knots.

Prediction mean	Contact mean	Filter mean	SPA	%Error	Obs. AOU	Pred. AO	Contact mean	Filtered mean
1.03	0.488	0.474	0.485	-2.26804	1	2.162	3.046	3.178
2.563	2.203	1.847	2.033	-9.14904	5	3.755	2.988	3.437
3.622	4.095	3.014	3.05	-1.18033	10	5.499	3	3.298
5.894	9.608	5.522	6.688	-17.4342	25	8.187	3.033	2.875
8.469	18.734	8.256	10.542	-21.6847	50	11.759	3.028	2.58
11.295	27.866	11.156	13.745	-18.8359	75	14.768	3.01	2.435
1.533	0.487	0.475	0.485	-2.06186	1	2.162	10.011	9.607
2.839	2.205	1.874	2.033	-7.82095	5	3.755	9.984	8.905
4	4.182	3.131	3.5	-10.5429	10	5.199	9.944	8.461
6.342	9.788	5.743	6.688	-14.1298	25	8.187	10.024	7.945

9.101	18.558	8.7	10.542	-17.473	50	11.759	10.05	7.625
11.972	27.983	11.653	13.745	-15.2201	75	14.768	9.981	7.512
3.513	0.51	0.513	0.485	5.773196	1	2.162	30.01	28.592
4.761	2.365	2.207	2.033	8.55878	5	3.755	30.038	26.093
6.158	4.456	3.791	3.5	8.314286	10	5.199	30.015	24.905
9.329	10.247	7.273	6.688	8.74701	25	8.187	29.992	23.28
13.198	19.553	11.421	10.542	8.338076	50	11.759	29.652	22.652
16.49	28.111	14.945	13.745	8.730447	75	14.768	30.026	22.335

Time interval between observations of .1 hours:

Standard Deviation of error in nautical miles:

Speed in knots.

Prediction mean	Contact mean	Filter mean	SPA	%Error	Obs. AOU	Pred. AO	Contact mean	Filtered mean
0.45	0.43	0.34	0.377	-9.81432	1	0.592	3.002	3.192
1.155	1.921	1.059	1.263	-16.152	5	1.504	3.021	3.081
1.689	3.728	1.621	1.995	-18.7469	10	2.22	3.003	2.847
2.084	5.502	2.034	2.573	-20.9483	15	2.78	2.995	2.734
0.495	0.429	0.348	0.377	-7.69231	1	0.592	10.003	9.568
1.241	1.934	1.112	1.263	-11.9557	5	1.504	10.005	8.966
1.761	3.748	1.665	1.995	-16.5414	10	2.22	10.026	8.711
2.221	5.549	2.144	2.573	-16.6731	15	2.78	10.015	8.563
0.778	0.466	0.428	0.377	13.52785	1	0.592	29.986	28.414
1.721	2.005	1.368	1.263	8.313539	5	1.504	29.998	26.532
2.408	3.832	2.104	1.995	5.463659	10	2.22	30.037	25.88
2.947	5.618	2.685	2.573	4.352895	15	2.78	30.008	25.47

Time interval between observations of .05 hours:

Standard Deviation of error in nautical miles:

Speed in knots.

Prediction mean	Contact mean	Filter mean	SPA	%Error	Obs. AOU	Pred. AO	Contact mean	Filtered mean
0.322	0.4	0.274	0.312	-12.1795	1	0.407	2.971	3.138
83	1.843	0.795	0.978	-18.7117	5	1.077	2.975	2.919
0.359	0.41	0.291	0.312	-6.73077	1	0.407	9.996	9.549
0.871	1.855	0.82	0.978	-16.1554	5	1.077	10.028	9.011
0.534	0.442	0.37	0.312	18.58974	1	0.407	29.997	28.449
1.176	1.908	1.045	0.978	6.850716	5	1.077	30.011	27.085

APPENDIX G: MODELS OF ACTUAL ESTIMATED FILTER OPERATION

This file simulates the basic operation of the ASSET tracker. A contact conducts a random tour at constant speed as a contact would in the simulation. Time interval between glimpses is an exponentially distributed variable. Statistics are collected to see how well the tracker tracks the target over the course of its path. The number of points plotted was adjusted to allow details to be seen.

$j := 0..75$ $\quad runtime := 48$ $\quad glimpse := .1$ $\quad \Delta_j := -\ln(rnd(1))\cdot glimpse$ $\quad mi := 4$ $\quad v := 10$ $\quad tm_0 := 0$ $\quad u := 10$ $\quad rng := u$

$tm_{j+1} := tm_j + \Delta_j$ $\quad N_j := until(runtime - tm_j, j)$ $\quad N2 := last(N) - 1$ $\quad t := 0..N2$ $\quad n := 0..N2+1$

$\Delta 1 := -\ln(rnd(1))\cdot mu$ $\quad \Delta 2 := -\ln(rnd(1))\cdot mu + \Delta 1$ $\quad \Delta 3 := -\ln(rnd(1))\cdot mu + \Delta 2$ $\quad \Delta 4 := -\ln(rnd(1))\cdot mi + \Delta 3$ $\quad \Delta 5 := -\ln(rnd(1))\cdot mi + \Delta 4$

$\Delta 6 := -\ln(rnd(1))\cdot mu + \Delta 5$ $\quad \Delta 7 := -\ln(rnd(1))\cdot mu + \Delta 6$ $\quad \Delta 8 := -\ln(rnd(1))\cdot mu + \Delta 7$ $\quad \Delta 9 := -\ln(rnd(1))\cdot mi + \Delta 8$ $\quad c1 := 2\cdot\pi\cdot rnd(1)$ $\quad c2 := 2\cdot\pi\cdot rnd(1)$

$c3 := 2\cdot\pi\cdot rnd(1)$ $\quad c4 := 2\cdot\pi\cdot rnd(1)$ $\quad c5 := 2\cdot\pi\cdot rnd(1)$ $\quad c6 := 2\cdot\pi\cdot rnd(1)$ $\quad c7 := 2\cdot\pi\cdot rnd(1)$ $\quad c8 := 2\cdot\pi\cdot rnd(1)$ $\quad c9 := 2\cdot\pi\cdot rnd(1)$ $\quad c10 := 2\cdot\pi\cdot rnd(1)$

$tc_n := if(tm_n < \Delta 1, c1, if(tm_n < \Delta 2, c2, if(tm_n < \Delta 3, c3, if(tm_n < \Delta 4, c4, if(tm_n < \Delta 5, c5, if(tm_n < \Delta 6, c6, if(tm_n < \Delta 7, c7, if(tm_n < \Delta 8, c8, if(tm_n < \Delta 9,$

$rand1_n := rnd(1)$ $\quad rand2_n := rnd(1)$ $\quad rand3_n := rnd(1)$ $\quad rand4_n := rnd(1)$

$xrng_n := \frac{rng}{2}\cdot\sqrt{-2\cdot\ln(rand1_n)}\cdot\cos(2\cdot\pi\cdot rand2_n)$ $\quad yrng_n := \frac{rng}{2}\cdot\sqrt{-2\cdot\ln(rand3_n)}\cdot\sin(2\cdot\pi\cdot rand4_n)$ $\quad rcse_n := mod\left(tc_n + rnd\left(\frac{\pi}{3}\right) - \frac{\pi}{6}, 2\cdot\pi\right)$

$rspd_n := v + (rnd(6) - 3)$ $\quad rbrg_n := 2\cdot\pi\cdot rnd(1)$ $\quad Yspd_n := v\cdot\sin(tc_n)$ $\quad Xspd_n := v\cdot\cos(tc_n)$ $\quad Ydis_0 := 0$ $\quad Xdis_0 := 0$

$$\begin{pmatrix} Ydis_{n+1} \\ Xdis_{n+1} \\ Yspd_n \\ Xspd_n \\ lat_n \\ lng_n \\ yrspd_n \\ xrspd_n \end{pmatrix} := \begin{pmatrix} Yspd_n\cdot\Delta_n + Ydis_n \\ Xspd_n\cdot\Delta_n + Xdis_n \\ v\cdot\sin(tc_n) \\ v\cdot\cos(tc_n) \\ yrng_n + Ydis_n \\ xrng_n + Xdis_n \\ rspd_n\cdot\sin(rcse_n) \\ rspd_n\cdot\cos(rcse_n) \end{pmatrix}$$

$\alpha := .25$ $\quad \sigma := \sqrt{\frac{100}{2}}$ $\quad r3 := \frac{\sigma^2}{2\cdot\alpha}$ $\quad r1 := .25\cdot u^2$ $\quad vl_0 := r1$ $\quad v2_0 := 0$ $\quad v3_0 := r3$ $\quad \phi 2_n := \frac{1}{\alpha}\cdot(1 - \exp(-\alpha\cdot\Delta_n))$

$\phi 3_n := \exp(-\alpha\cdot\Delta_n)$ $\quad q1_n := \frac{\sigma^2}{\alpha^2}\cdot\left(\Delta_n - \left(\frac{1}{\alpha}\right)\cdot(2\cdot(1 - \exp(-\alpha\cdot\Delta_n)) - .5\cdot(1 - \exp(-\alpha\cdot 2\cdot\Delta_n)))\right)$

$q2_n := \frac{\sigma^2}{\alpha^2}\cdot(.5 - \exp(-\alpha\cdot\Delta_n) + .5\cdot\exp(-\alpha\cdot 2\cdot\Delta_n))$ $\quad q3_n := .5\cdot\frac{\sigma^2}{\alpha}\cdot(1 - \exp(-\alpha\cdot 2\cdot\Delta_n))$

$$\begin{pmatrix} vl_{n+1} \\ v2_{n+1} \\ v3_{n+1} \end{pmatrix} := \begin{pmatrix} v2_n + \phi 2_n\cdot v3_n \\ \phi 3_n\cdot v3_n + q2_n \\ r1\cdot\dfrac{(vl_n + 2\cdot\phi 2_n\cdot v2_n + \phi 2_n^2\cdot v3_n + q1_n)\cdot(\phi 3_n^2\cdot v3_n + q3_n + r3) - (v2_n + \phi 2_n\cdot v3_n)\cdot(\phi 3_n + q2_n)^2}{(vl_n + 2\cdot\phi 2_n\cdot v2_n + \phi 2_n^2\cdot v3_n + q1_n + r1)\cdot(\phi 3_n^2\cdot v3_n + q3_n + r3) - (v2_n + \phi 2_n\cdot v3_n)\cdot(\phi 3_n + q2_n)^2} \\ r3\cdot\dfrac{(vl_n + 2\cdot\phi 2_n\cdot v2_n + \phi 2_n^2\cdot v3_n + q1_n + r1)\cdot(\phi 3_n^2\cdot v3_n + q3_n) - (v2_n + \phi 2_n\cdot v3_n)\cdot(\phi 3_n + q2_n)^2}{(vl_n + 2\cdot\phi 2_n\cdot v2_n + \phi 2_n^2\cdot v3_n + q1_n + r1)\cdot(\phi 3_n^2\cdot v3_n + q3_n + r3) - (v2_n + \phi 2_n\cdot v3_n)\cdot(\phi 3_n + q2_n)^2} \end{pmatrix}$$

$k1_n := \frac{vl_n}{r1}$ $\quad k2a_n := \frac{v2_n}{r3}$ $\quad k2b_n := \frac{v2_n}{r1}$ $\quad k3_n := \frac{v3_n}{r3}$ $\quad Vx_0 := xrspd_0$ $\quad Vy_0 := yrspd_0$ $\quad X_0 := lng_0$ $\quad Y_0 := lat_0$ $\quad mv2 := mean(v2)$ $\quad mv3 := mean(v3)$

$$\begin{bmatrix} Y_{t+1} \\ X_{t+1} \\ Vy_{t+1} \\ Vx_{t+1} \end{bmatrix} = \begin{bmatrix} \frac{Y_t \cdot \alpha + Vy_t - Vy_t \cdot \exp(-\alpha \cdot \Delta_t)}{\alpha} + k1_{t+1} \cdot \left(lat_{t+1} - \frac{Y_t \cdot \alpha + Vy_t - Vy_t \cdot \exp(-\alpha \cdot \Delta_t)}{\alpha} \right) + k2a_{t+1} \cdot (yrspd_{t+1} - Vy_t \cdot \exp(-\alpha \cdot \Delta_t)) \\ \frac{X_t \cdot \alpha + Vx_t - Vx_t \cdot \exp(-\alpha \cdot \Delta_t)}{\alpha} + k1_{t+1} \cdot \left(lng_{t+1} - \frac{X_t \cdot \alpha + Vx_t - Vx_t \cdot \exp(-\alpha \cdot \Delta_t)}{\alpha} \right) + k2a_{t+1} \cdot (xrspd_{t+1} - Vx_t \cdot \exp(-\alpha \cdot \Delta_t)) \\ Vy_t \cdot \exp(-\alpha \cdot \Delta_t) + k2b_{t+1} \cdot \left(lat_{t+1} - \frac{Y_t \cdot \alpha + Vy_t - Vy_t \cdot \exp(-\alpha \cdot \Delta_t)}{\alpha} \right) + k3_{t+1} \cdot (yrspd_{t+1} - Vy_t \cdot \exp(-\alpha \cdot \Delta_t)) \\ Vx_t \cdot \exp(-\alpha \cdot \Delta_t) + k2b_{t+1} \cdot \left(lng_{t+1} - \frac{X_t \cdot \alpha + Vx_t - Vx_t \cdot \exp(-\alpha \cdot \Delta_t)}{\alpha} \right) + k3_{t+1} \cdot (xrspd_{t+1} - Vx_t \cdot \exp(-\alpha \cdot \Delta_t)) \end{bmatrix}$$

$$\begin{bmatrix} Viou1_{t+1} \\ Viou2_{t+1} \\ Viou3_{t+1} \end{bmatrix} = \begin{bmatrix} v1_t + 2 \cdot \phi 2_t \cdot v2_t + \phi 2_t^2 \cdot v3_t + q1_t \\ v2_t + \phi 2_t \cdot v3_t \cdot \phi 3_t + q2_t \\ \phi 3_t^2 \cdot v3_t + q3_t \end{bmatrix} \qquad \begin{bmatrix} Xiou_{t+1} \\ Yiou_{t+1} \\ Vxiou_{t+1} \\ Vyiou_{t+1} \end{bmatrix} = \begin{bmatrix} X_t + \frac{1}{\alpha} \cdot (1 - \exp(-\alpha \cdot \Delta_t)) \cdot Vx_t \\ Y_t + \frac{1}{\alpha} \cdot (1 - \exp(-\alpha \cdot \Delta_t)) \cdot Vy_t \\ \exp(-\alpha \cdot \Delta_t) \cdot Vx_t \\ \exp(-\alpha \cdot \Delta_t) \cdot Vy_t \end{bmatrix}$$

Parameters for this run:

$D = mean(\Delta)$ $N2 = 49$ $\sigma = 7.071068$ $\alpha = 0.25$ $u = 5$

$AOU = \sqrt{mean(v1)}$ $IOU = \sqrt{mean(Viou1)}$ $AOU = 1.087788$ $IOU = 1.197159$

Determine the difference between the true target position, and the filtered position:

$Fx_t = X_t - Xdis_t$ $Fy_t = Y_t - Ydis_t$

the true target position, and the contact position and the true target position, and the IOU-predicted position:

$Cx_t = lng_t - Xdis_t$ $Cy_t = lat_t - Ydis_t$ $Ix_t = Xiou_t - Xdis_t$ $Iy_t = Yiou_t - Ydis_t$

The root-mean-square error is found for the data:

$F2x_t = (Fx_t)^2$ $F2y_t = (Fy_t)^2$ $C2x_t = (Cx_t)^2$ $C2y_t = (Cy_t)^2$ $I2x_t = (Ix_t)^2$ $I2y_t = (Iy_t)^2$

$SFx = \Sigma F2x$ $SFy = \Sigma F2y$ $SCx = \Sigma C2x$ $SCy = \Sigma F2y$ $SIx = \Sigma I2x$ $SIy = \Sigma I2y$

$Ferr = \sqrt{\left(\frac{SFx}{last(Fx)} + \frac{SFy}{last(Fy)}\right) \cdot .5}$ $Cerr = \sqrt{\left(\frac{SCx}{last(Cx)} + \frac{SCy}{last(Cy)}\right) \cdot .5}$ $Ierr = \sqrt{\left(\frac{SIx}{last(Ix)} + \frac{SIy}{last(Iy)}\right) \cdot .5}$ $Ferr = 0.952507$ $Ierr = 1.01829$

$Cerr = 1.740496$

The mean filtered and contact velocities is found:

$FV_t = \sqrt{(Vx_t)^2 + (Vy_t)^2}$ $CV_t = \sqrt{(xrspd_t)^2 + (yrspd_t)^2}$ $mean(FV) = 8.574437$ $mean(CV) = 10.102023$

$mFV = mean(FV)$

$stdev(FV) = 0.773034$ $stdev(CV) = 1.641369$ $mCV = mean(CV)$

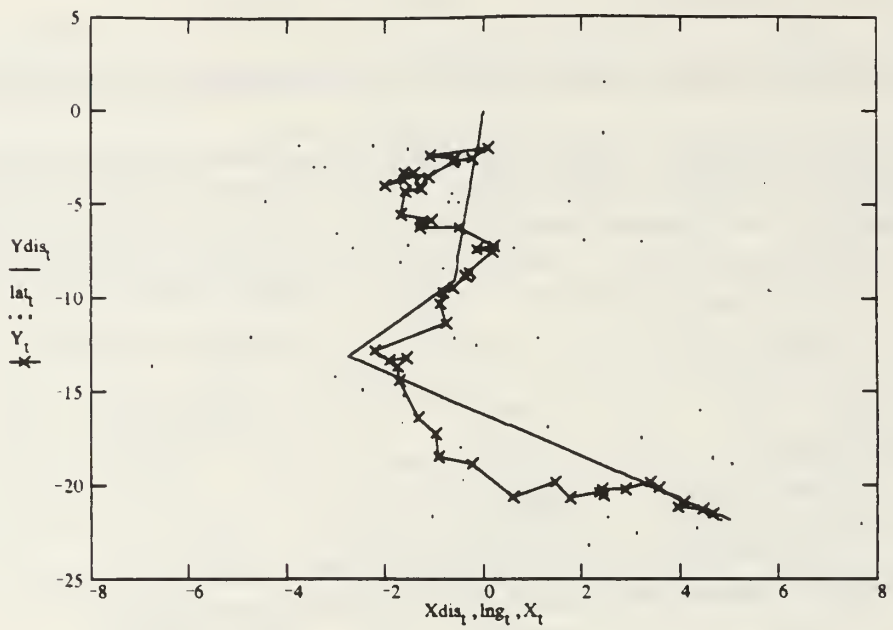
The RMS error in filtered position is compared to the RMS error in the contact position:

$TR = Cerr - Ferr$ $TR = 0.787989$

The difference from the computed error is then computed for the current run:

$COMPerr = AOU - Ferr$ $COMPerr = 0.135281$

Filtered track using the actual algorithm, with mean glimpse interval 0.05 hour and Sensor AOU = 5nm:



$\Delta 1 = 0.896146$
$\Delta 2 = 1.251034$
$\Delta 3 = 3.394831$
$\Delta 4 = 4.572997$
$\Delta 5 = 6.685208$
$\Delta 6 = 16.686341$
$\Delta 7 = 31.313094$
$\Delta 8 = 31.611555$
$\Delta 9 = 34.12239$

$N2 = 49$

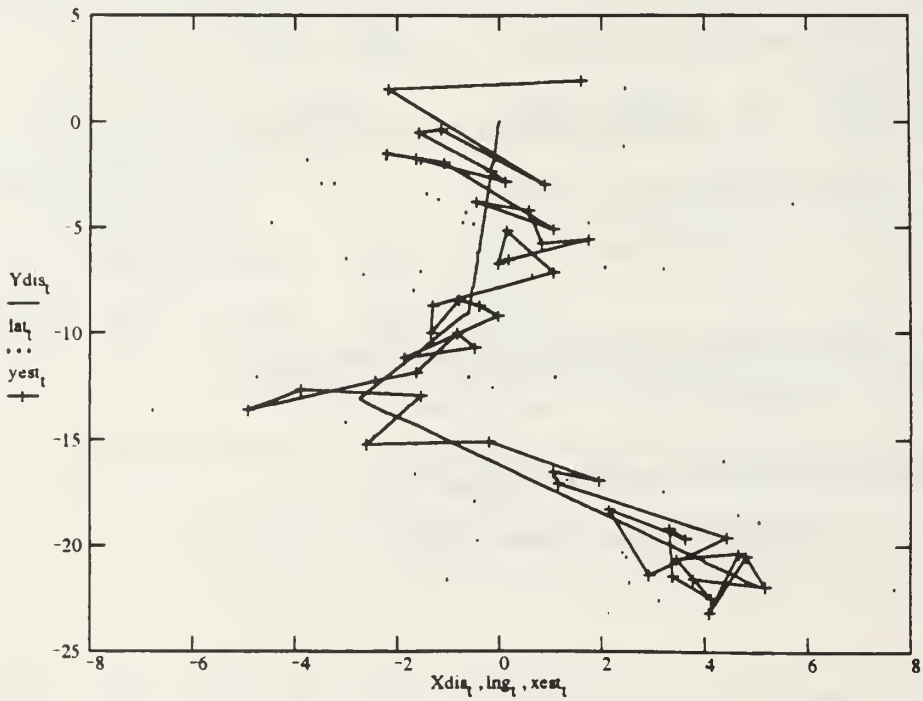
$xest_0 = 0$ $yest_0 = 0$

$xest_n = \sqrt{vl_n} \cdot \sqrt{(2 \cdot \ln(rnd(1)))} \cdot \cos(2 \cdot \pi \cdot rnd(1)) + Xdis_n$ $yest_n = \sqrt{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \sin(2 \cdot \pi \cdot rnd(1)) + Ydis_n$

$xVe_t = Vx_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + (xrspd_t + 1 - Vx_t \cdot \exp(-\alpha \cdot \Delta_t))$ $yVe_t = Vy_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + (yrspd_t + 1 - Vy_t \cdot \exp(-\alpha \cdot \Delta_t))$

$Vdiffx_t = xVe_t - Vx_t$ $mean(Vdiffx) = 0.133526$ $Vdiffy_t = yVe_t - Vy_t$ $mean(Vdiffy) = 0.030425$

Filtered track using estimation technique (using full noise values):



$Estx_t = xest_t - Xdis_t$

$Esty_t = yest_t - Ydis_t$

$Ex_t = Estx_t^2$ $Ey_t = Esty_t^2$

$SEx = \Sigma Ex$ $SEy = \Sigma Ey$

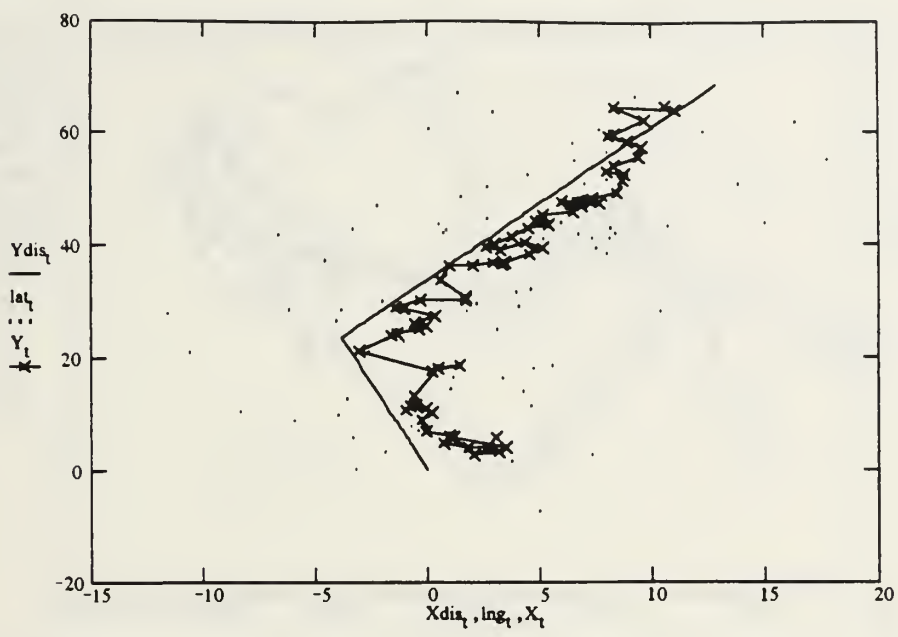
$Eerr = \left(\frac{(SEx)}{last(Ex)} + \frac{(SEy)}{last(Ey)}\right)^{.5}$

$Eerr = 1.028216$

$Ferr = 0.952507$

$\sqrt{mean(vl)} = 1.087788$

Filtered track using the actual algorithm, with mean glimpse interval 0.1 hours and Sensor AOU = 10nm:



$\Delta 1 = 2.089507$
$\Delta 2 = 10.470267$
$\Delta 3 = 14.93814$
$\Delta 4 = 17.572047$
$\Delta 5 = 20.489058$
$\Delta 6 = 22.232939$
$\Delta 7 = 25.150709$
$\Delta 8 = 25.233842$
$\Delta 9 = 45.284693$

$N2 = 74$

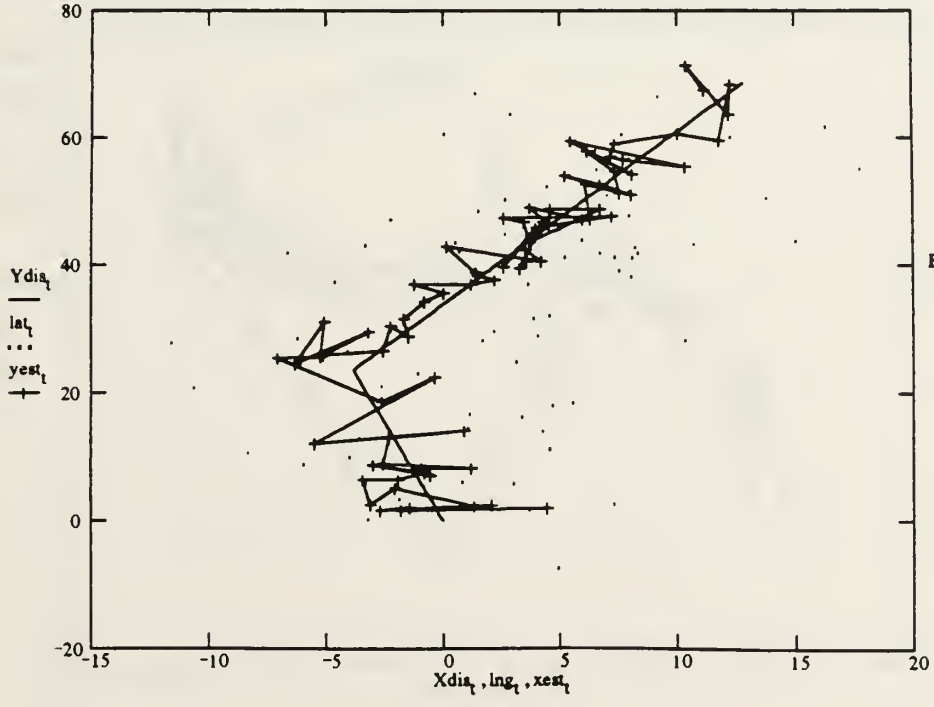
$xest_0 = 0$ $yest_0 = 0$

$xest_n = \sqrt{v1_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \cos[2 \cdot \pi \cdot rnd(1)] + Xdis_n$ $yest_n = \sqrt{v1_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \sin[2 \cdot \pi \cdot rnd(1)] + Ydis_n$

$xVe_t = Vx_t \cdot \exp[-\alpha \cdot \Delta_t] + k3_t + H \cdot [xrspd_t + H - Vx_t \cdot \exp[-\alpha \cdot \Delta_t]]$ $yVe_t = Vy_t \cdot \exp[-\alpha \cdot \Delta_t] + k3_t + H \cdot [yrspd_t + H - Vy_t \cdot \exp[-\alpha \cdot \Delta_t]]$

$Vdiffx_t = xVe_t - Vx_t$ $mean(Vdiffx) = 0.056692$ $Vdiffy_t = yVe_t - Vy_t$ $mean(Vdiffy) = 0.013961$

Filtered track using estimation technique (using full noise values):



$Estx_t = xest_t - Xdis_t$

$Esty_t = yest_t - Ydis_t$

$Ex_t = [Estx_t]^2$ $Ey_t = [Esty_t]^2$

$SEx = \Sigma Ex$ $SEy = \Sigma Ey$

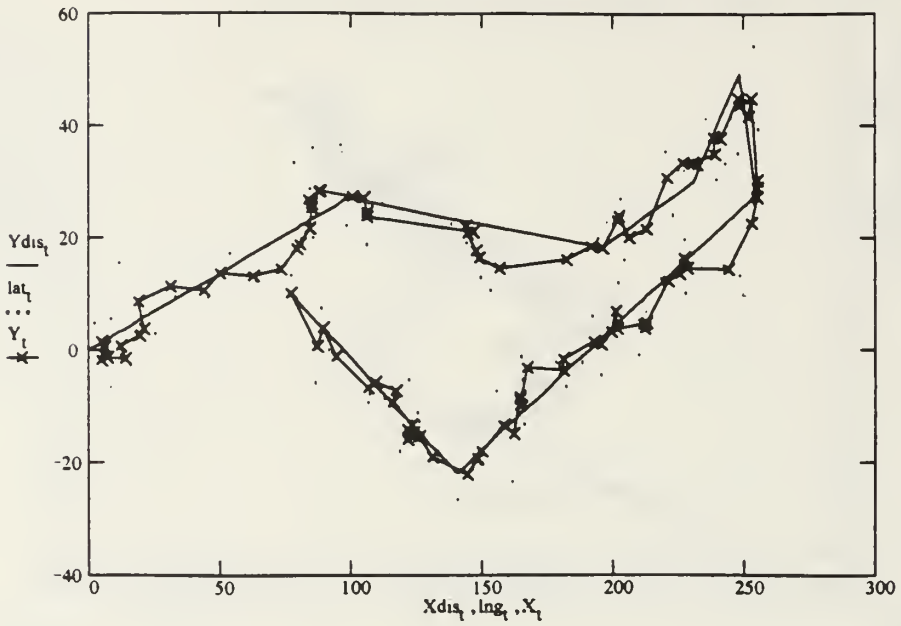
$Eerr = \sqrt{\left[\frac{(SEx)}{last(Ex)} + \frac{(SEy)}{last(Ey)}\right]} \cdot .5$

$Eerr = 1.925021$

$Ferr = 2.165015$

$\sqrt{mean(v1)} = 2.072638$

Filtered track using the actual algorithm, with mean glimpse interval 0.5 hours and Sensor AOU = 10nm:



$\Delta 1 = 9.779778$
$\Delta 2 = 19.746842$
$\Delta 3 = 23.257026$
$\Delta 4 = 23.336796$
$\Delta 5 = 25.985588$
$\Delta 6 = 26.130136$
$\Delta 7 = 27.402486$
$\Delta 8 = 28.369044$
$\Delta 9 = 40.739811$

$N2 = 91$

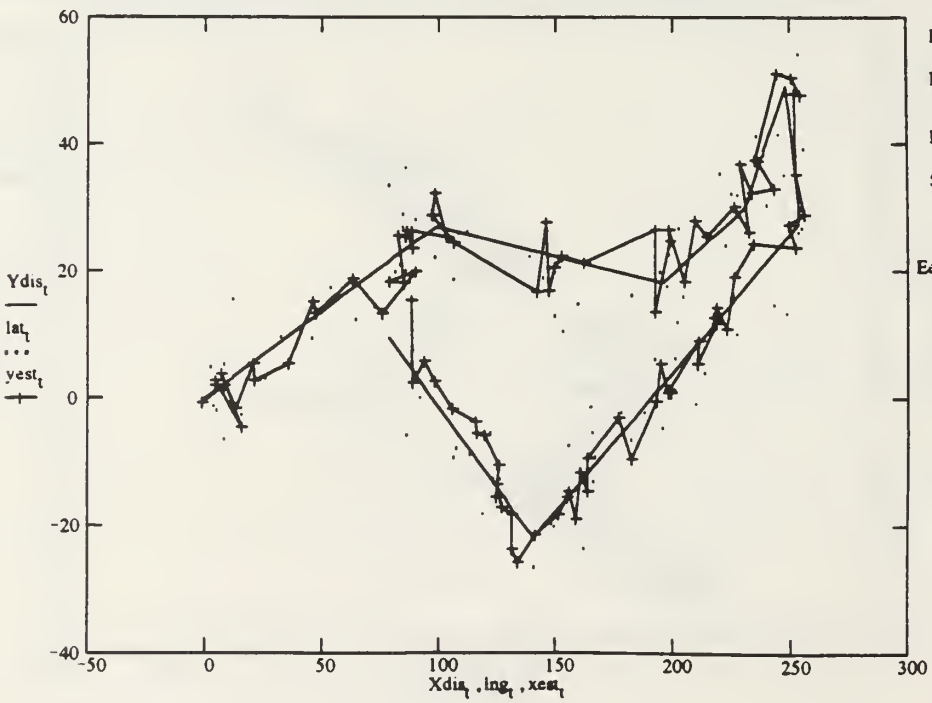
$xest_0 = 0$ $yest_0 = 0$

$xest_n = \sqrt{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1))) \cdot \cos(2 \cdot \pi \cdot rnd(1))} + Xdis_n$ $yest_n = \sqrt{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1))) \cdot \sin(2 \cdot \pi \cdot rnd(1))} + Ydis_n$

$xVe_t = Vx_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + (xrspd_t + (-Vx_t \cdot \exp(-\alpha \cdot \Delta_t)))$ $yVe_t = Vy_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + (yrspd_t + (-Vy_t \cdot \exp(-\alpha \cdot \Delta_t)))$

$Vdiffx_t = xVe_t - Vx_t$ $mean(Vdiffx) = -0.104991$ $Vdiffy_t = yVe_t - Vy_t$ $mean(Vdiffy) = 0.040473$

Filtered track using estimation technique (using full noise values):



$Estx_t = xest_t - Xdis_t$

$Esty_t = yest_t - Ydis_t$

$Ex_t = (Estx_t)^2$ $Ey_t = (Esty_t)^2$

$SEx = \Sigma Ex$ $SEy = \Sigma Ey$

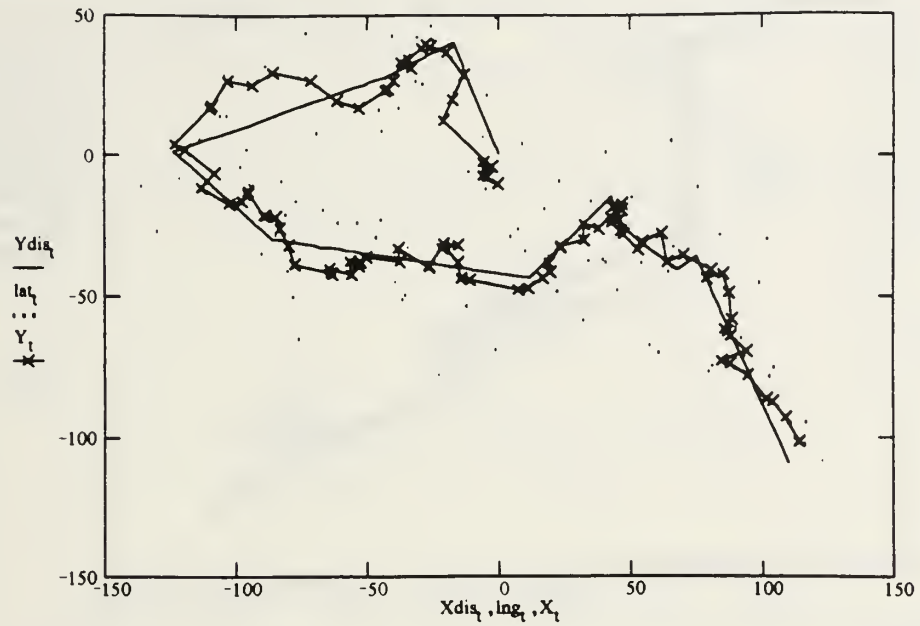
$Eerr = \sqrt{\left(\frac{(SEx)}{last(Ex)} + \frac{(SEy)}{last(Ey)}\right) \cdot .5}$

$Eerr = 3.402076$

$Ferr = 2.944767$

$\sqrt{mean(vl)} = 3.398984$

Filtered track using the actual algorithm, with mean glimpse interval 0.5 hours and Sensor AOU = 30nm:



$\Delta 1 = 3.915512$
$\Delta 2 = 6.91616$
$\Delta 3 = 15.287011$
$\Delta 4 = 20.200269$
$\Delta 5 = 30.204378$
$\Delta 6 = 34.696944$
$\Delta 7 = 35.833538$
$\Delta 8 = 38.376747$
$\Delta 9 = 38.669162$

$N2 = 96$

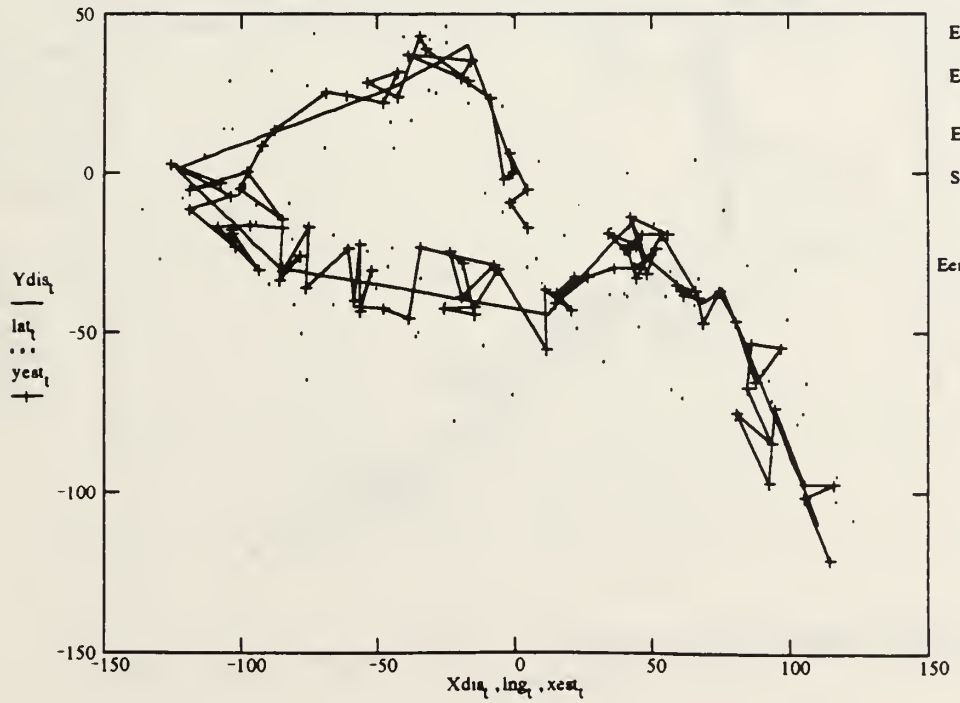
$xest_0 = 0$ $yest_0 = 0$

$xest_n = \sqrt{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \cos(2 \cdot \pi \cdot rnd(1)) + Xdis_n$ $yest_n = \sqrt{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \sin(2 \cdot \pi \cdot rnd(1)) + Ydis_n$

$xVe_t = Vx_t \cdot \exp(-\alpha \cdot \Delta_t) + tk3_t + H \cdot (xrspd_t + H - Vx_t \cdot \exp(-\alpha \cdot \Delta_t))$ $yVe_t = Vy_t \cdot \exp(-\alpha \cdot \Delta_t) + tk3_t + H \cdot (yrspd_t + H - Vy_t \cdot \exp(-\alpha \cdot \Delta_t))$

$Vdiffx_t = xVe_t - Vx_t$ $mean(Vdiffx) = -0.056681$ $Vdiffy_t = yVe_t - Vy_t$ $mean(Vdiffy) = -0.145247$

Filtered track using estimation technique (using full noise values):



$Estx_t = xest_t - Xdis_t$

$Esty_t = yest_t - Ydis_t$

$Ex_t = Estx_t^2$ $Ey_t = Esty_t^2$

$SEx = \sum Ex$ $SEy = \sum Ey$

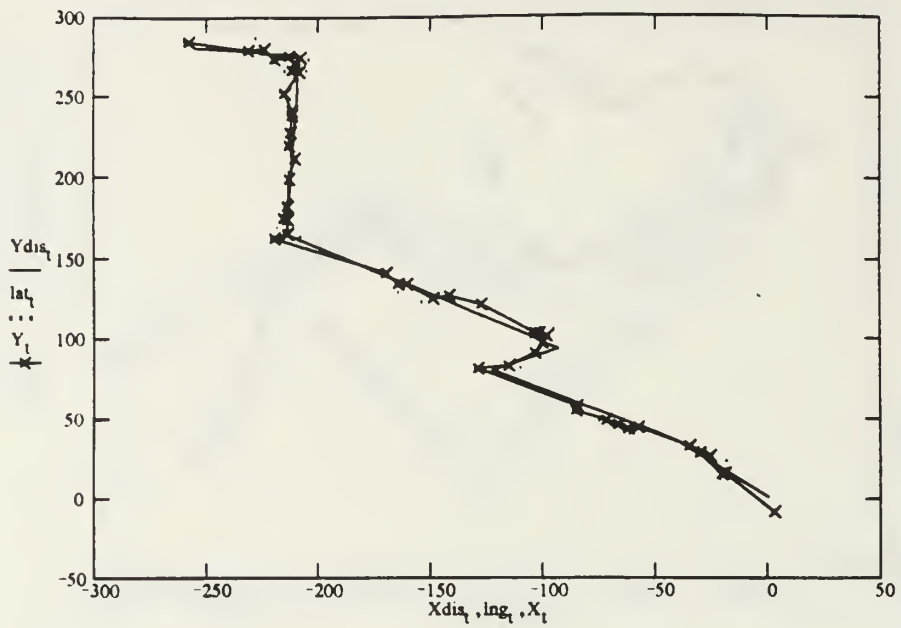
$Eerr = \sqrt{\left(\frac{(SEx)}{last(Ex)} + \frac{(SEy)}{last(Ey)}\right)} \cdot .5$

$Eerr = 7.774309$

$Ferr = 5.817649$

$\sqrt{mean(vl)} = 7.596565$

Filtered track using the actual algorithm, with mean glimpse interval 1 hour and Sensor AOU = 10nm:



$\Delta 1 = 5.022727$
$\Delta 2 = 12.437775$
$\Delta 3 = 17.876472$
$\Delta 4 = 28.662312$
$\Delta 5 = 42.644194$
$\Delta 6 = 47.842995$
$\Delta 7 = 49.239743$
$\Delta 8 = 50.274019$
$\Delta 9 = 53.491883$

$N2 = 45$

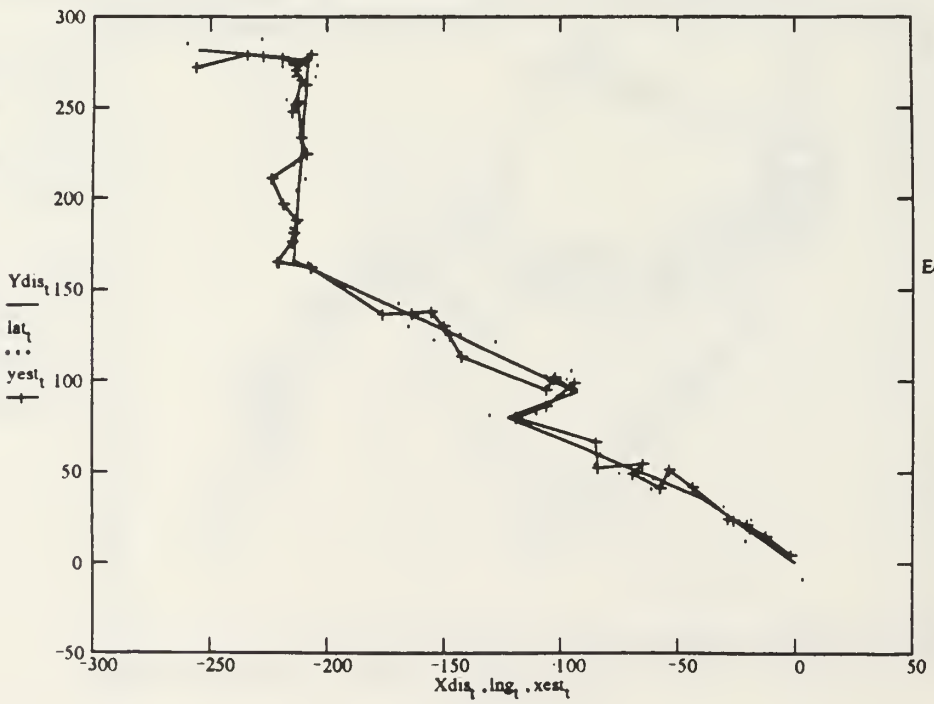
$xest_0 = 0$ $yest_0 = 0$

$$xest_n = \sqrt{vl_n}\cdot\sqrt{(-2\cdot\ln(rnd(1)))}\cdot\cos(2\cdot\pi\cdot rnd(1)) + Xdis_n \qquad yest_n = \sqrt{vl_n}\cdot\sqrt{(-2\cdot\ln(rnd(1)))}\cdot\sin(2\cdot\pi\cdot rnd(1)) + Ydis_n$$

$$xVe_t = Vx_t\cdot\exp(-\alpha\cdot\Delta_t) + k3_t + 1\cdot(xrspd_t + 1 - Vx_t\cdot\exp(-\alpha\cdot\Delta_t)) \qquad yVe_t = Vy_t\cdot\exp(-\alpha\cdot\Delta_t) + k3_t + 1\cdot(yrspd_t + 1 - Vy_t\cdot\exp(-\alpha\cdot\Delta_t))$$

$Vdiffx_t = xVe_t - Vx_t$ $mean(Vdiffx) = 0.472115$ $Vdiffy_t = yVe_t - Vy_t$ $mean(Vdiffy) = -0.66914$

Filtered track using estimation technique (using full noise values):



$Estx_t = xest_t - Xdis_t$

$Esty_t = yest_t - Ydis_t$

$Ex_t = Estx_t^2$ $Ey_t = Esty_t^2$

$SEx = \Sigma Ex$ $SEy = \Sigma Ey$

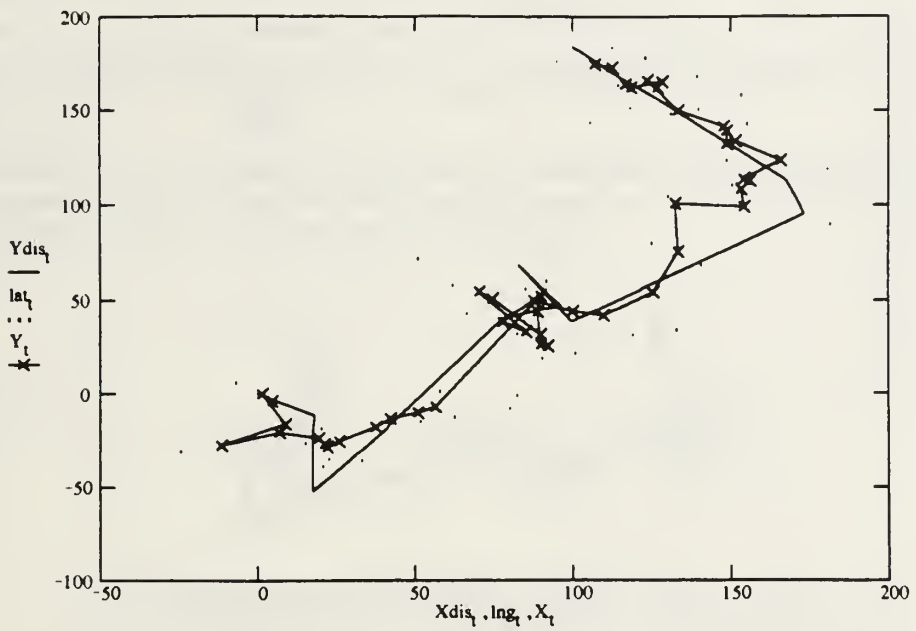
$$Eerr = \sqrt{\left(\frac{(SEx)}{last(Ex)} + \frac{(SEy)}{last(Ey)}\right)}\cdot .5$$

$Eerr = 4.03991$

$Ferr = 3.395647$

$\sqrt{mean(vl)} = 3.978631$

Filtered track using the actual algorithm, with mean glimpse interval 1 hour and Sensor AOU = 30nm:



$\Delta 1 = 0.155067$
$\Delta 2 = 4.15131$
$\Delta 3 = 9.030112$
$\Delta 4 = 14.972823$
$\Delta 5 = 19.212$
$\Delta 6 = 21.285281$
$\Delta 7 = 24.284637$
$\Delta 8 = 33.048225$
$\Delta 9 = 36.179972$

$N2 = 44$

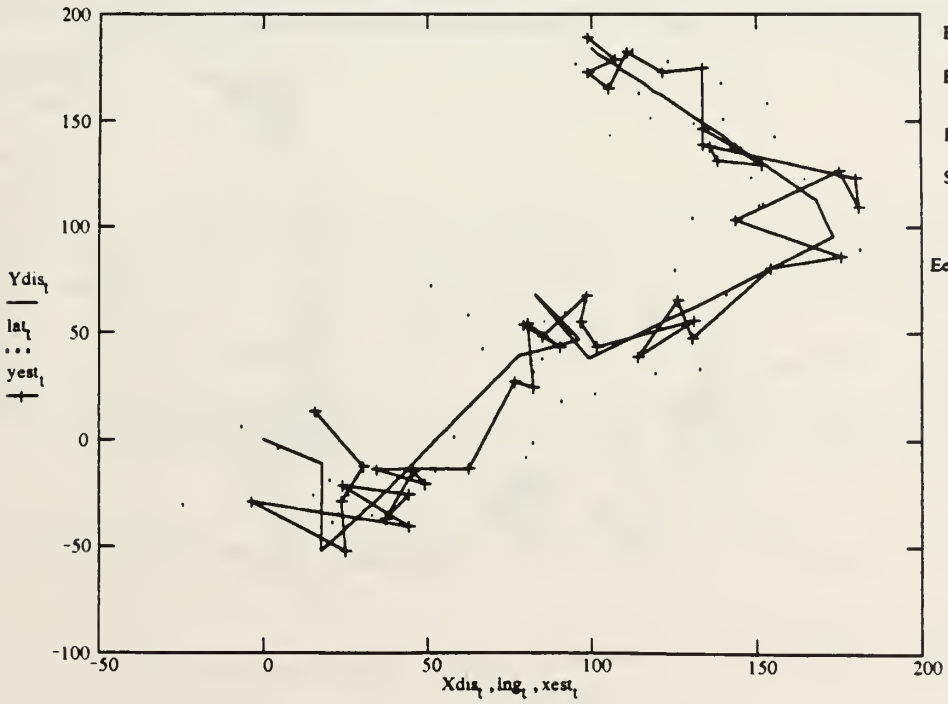
$xest_0 = 0$ $yest_0 = 0$

$xest_n = \overline{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \cos(2 \cdot \pi \cdot rnd(1)) + Xdis_n$ $yest_n = \overline{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \sin(2 \cdot \pi \cdot rnd(1)) + Ydis_n$

$xVe_t = Vx_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + 1 \cdot (xrspd_t + 1 - Vx_t \cdot \exp(-\alpha \cdot \Delta_t))$ $yVe_t = Vy_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + 1 \cdot (yrspd_t + 1 - Vy_t \cdot \exp(-\alpha \cdot \Delta_t))$

$Vdiffx_t = xVe_t - Vx_t$ $mean(Vdiffx) = -0.495172$ $Vdiffy_t = yVe_t - Vy_t$ $mean(Vdiffy) = 0.01928$

Filtered track using estimation technique (using full noise values):



$Estx_t = xest_t - Xdis_t$

$Esty_t = yest_t - Ydis_t$

$Ex_t = Estx_t^2$ $Ey_t = Esty_t^2$

$SEx = \sum Ex$ $SEy = \sum Ey$

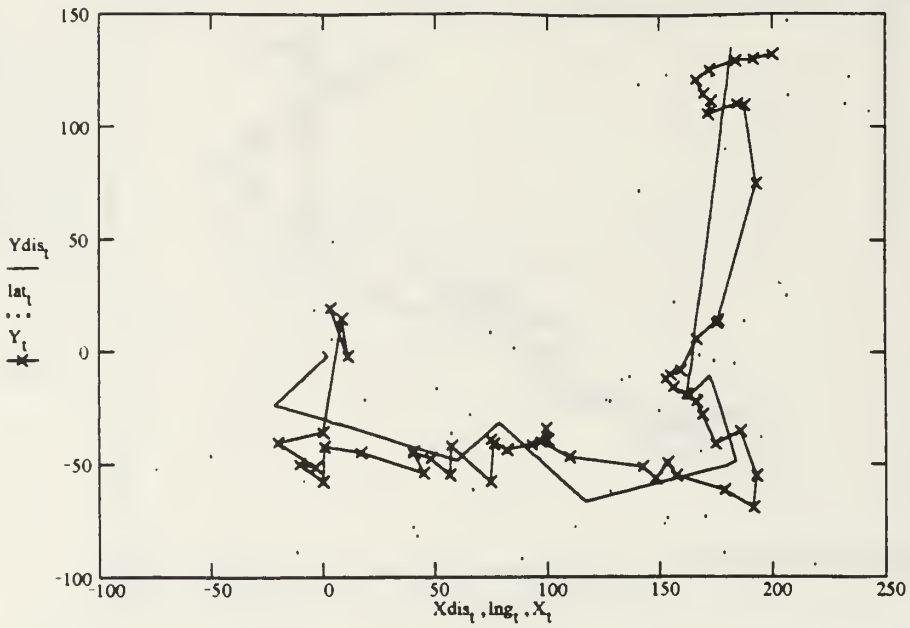
$Eerr = \left(\frac{(SEx)}{last(Ex)} + \frac{(SEy)}{last(Ey)}\right)^{.5}$

$Eerr = 9.81617$

$Ferr = 12.186325$

$\overline{mean(vl)} = 9.549774$

Filtered track using the actual algorithm, with mean glimpse interval 1 hour and Sensor AOU = 60nm:



$\Delta 1 = 0.003175$
$\Delta 2 = 3.366652$
$\Delta 3 = 11.418094$
$\Delta 4 = 14.422879$
$\Delta 5 = 19.245857$
$\Delta 6 = 24.664634$
$\Delta 7 = 26.370376$
$\Delta 8 = 30.333605$
$\Delta 9 = 30.916577$

$N2 = 54$

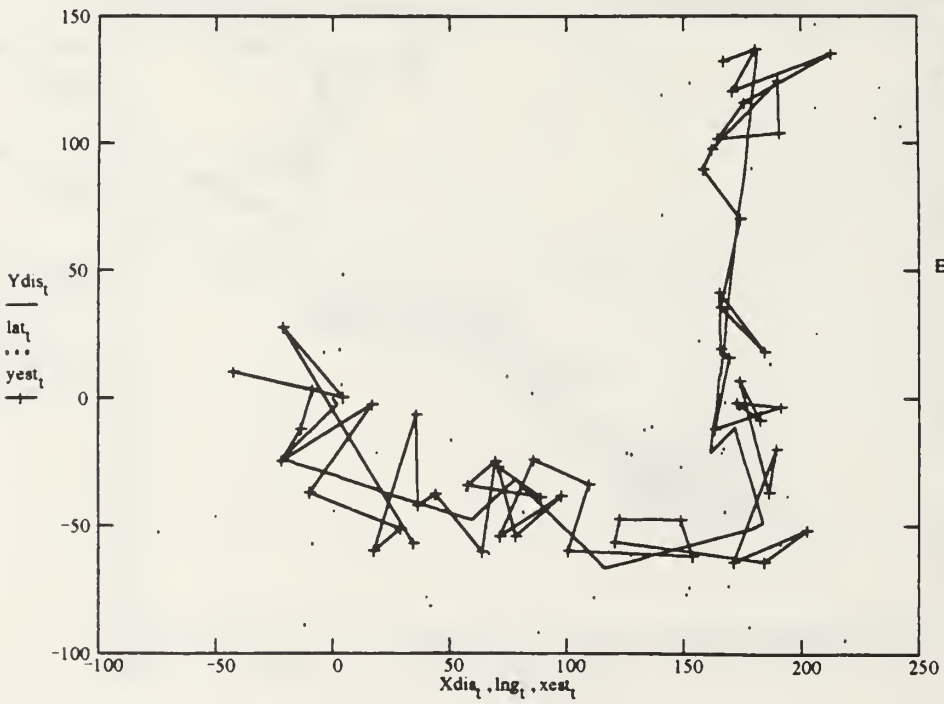
$xest_0 = 0$ $yest_0 = 0$

$xest_n = \sqrt{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \cos(2 \cdot \pi \cdot rnd(1)) + Xdis_n$ $yest_n = \sqrt{vl_n} \cdot \sqrt{(-2 \cdot \ln(rnd(1)))} \cdot \sin(2 \cdot \pi \cdot rnd(1)) + Ydis_n$

$xVe_t = Vx_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + (xrspd_t + 1 - Vx_t \cdot \exp(-\alpha \cdot \Delta_t))$ $yVe_t = Vy_t \cdot \exp(-\alpha \cdot \Delta_t) + k3_t + (yrspd_t + 1 - Vy_t \cdot \exp(-\alpha \cdot \Delta_t))$

$Vdiffx_t = xVe_t - Vx_t$ $mean(Vdiffx) = -0.202094$ $Vdiffy_t = yVe_t - Vy_t$ $mean(Vdiffy) = 0.294623$

Filtered track using estimation technique (using full noise values):



$Estx_t = xest_t - Xdis_t$

$Esty_t = yest_t - Ydis_t$

$Ex_t = Estx_t^2$ $Ey_t = Esty_t^2$

$SEx = \sum Ex$ $SEy = \sum Ey$

$Eerr = \sqrt{\left(\frac{(SEx)}{last(Ex)} + \frac{(SEy)}{last(Ey)}\right)} \cdot .5$

$Eerr = 15.785448$

$Ferr = 12.430599$

$\sqrt{mean(vl)} = 15.165514$

APPENDIX H: KALMAN GAIN VERSUS SENSOR AOU SIZE

This appendix shows the relationship between the steady state Kalman gain Ka and the Sensor AOU size. The impact of using the the approximations from Appendix A is also illustrated. The data for each case was written to a seperate file and the graph alone displayed

$n := 0..25 \qquad \Delta := .1 \qquad u := 1000 \qquad \alpha := .25$

$\sigma := \sqrt{50} \qquad r3 := \frac{\sigma^2}{2\cdot\alpha} \qquad r1 := .25\cdot u^2$

$v1e_0 := r1 \qquad v2e_0 := 0 \qquad v3e_0 := r3 \qquad v1_0 := r1 \qquad v2_0 := 0 \qquad v3_0 := r3$

$\phi 2 := \frac{1}{\alpha}\cdot\left[1 - \exp\left[-\alpha\cdot\Delta\right]\right] \qquad \phi 3 := \exp\left[-\alpha\cdot\Delta\right]$

$q1 := \frac{\sigma^2}{\alpha^2}\cdot\left[\Delta - \frac{1}{\alpha}\cdot\left[2\cdot\left[1 - \exp\left[-\alpha\cdot\Delta\right]\right] - .5\cdot\left[1 - \exp\left[-\alpha\cdot 2\cdot\Delta\right]\right]\right]\right]$

$q2 := \frac{\sigma^2}{\alpha^2}\cdot\left[.5 - \exp\left[-\alpha\cdot\Delta\right] + .5\cdot\exp\left[-\alpha\cdot 2\cdot\Delta\right]\right]$

$q3 := .5\cdot\frac{\sigma^2}{\alpha}\cdot\left[1 - \exp\left[-\alpha\cdot 2\cdot\Delta\right]\right]$

$$\begin{bmatrix} v1_{n+1} \\ v2_{n+1} \\ v3_{n+1} \end{bmatrix} = \begin{bmatrix} r1\cdot\dfrac{v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1}{v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1 + r1} \\ \dfrac{\left[v2_n + \phi 2\cdot v3_n\right]\cdot\phi 3 + q2}{v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1 + r1}\cdot r1\cdot r3 \\ r3\cdot\dfrac{v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1 + r1}{v1_n + 2\cdot\phi 2\cdot v2_n + \phi 2^2\cdot v3_n + q1 + r1} \end{bmatrix}$$

Using the transition and process noise approximations from Appendix A to simplify the calculations:

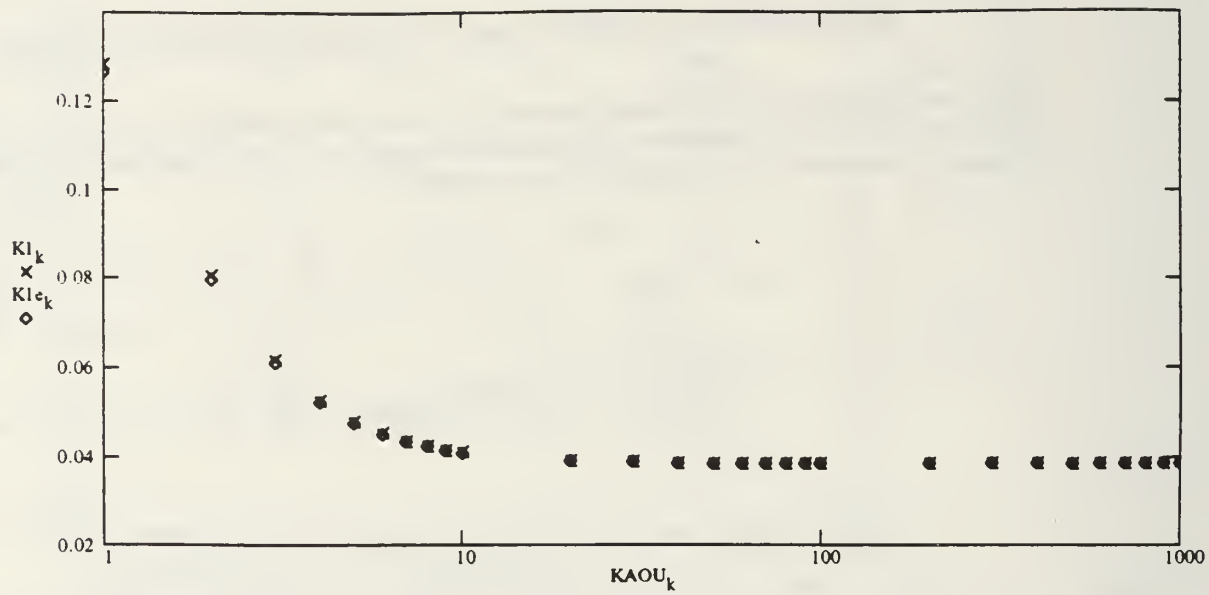
$\phi 2\varepsilon := \frac{1}{2}\cdot\Delta\cdot\left[\alpha\cdot\Delta - 2\right] \qquad \phi 3\varepsilon := 1 + \frac{1}{2}\cdot\alpha\cdot\Delta\cdot\left[\alpha\cdot\Delta - 2\right] \qquad q1e := \sigma^2\cdot\left[\Delta\right]^3\cdot\left[\frac{1}{3} - \frac{1}{4}\cdot\alpha\cdot\Delta\right] \qquad q2e := \sigma^2\cdot\frac{1}{2}\cdot\left[\Delta\right]^2\cdot\left[1 - \alpha\cdot\Delta\right]$

$q3e := \sigma^2\cdot\Delta\cdot\left[1 - \alpha\cdot\Delta\right]$

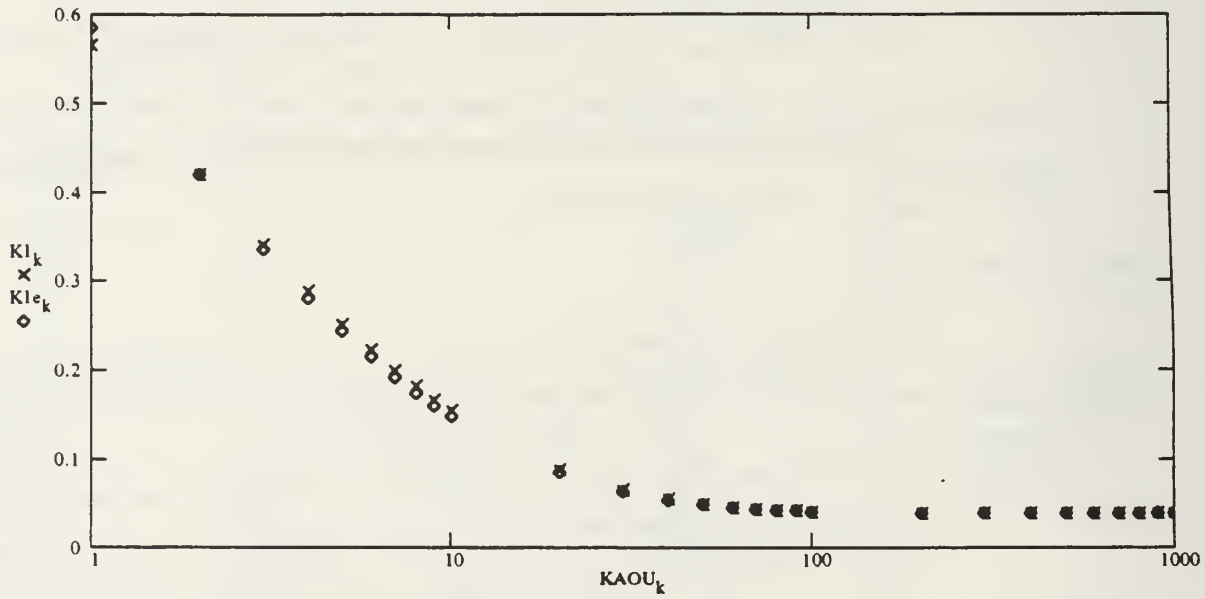
$$\begin{bmatrix} v1e_{n+1} \\ v2e_{n+1} \\ v3e_{n+1} \end{bmatrix} = \begin{bmatrix} r1\cdot\dfrac{v1e_n + 2\cdot\phi 2\varepsilon\cdot v2e_n + \phi 2\varepsilon^2\cdot v3e_n + q1e}{v1e_n + 2\cdot\phi 2\varepsilon\cdot v2e_n + \phi 2\varepsilon^2\cdot v3e_n + q1e + r1} \\ \dfrac{\left[v2e_n + \phi 2\varepsilon\cdot v3e_n\right]\cdot\phi 3\varepsilon + q2e}{v1e_n + 2\cdot\phi 2\varepsilon\cdot v2e_n + \phi 2\varepsilon^2\cdot v3e_n + q1e + r1}\cdot r1\cdot r3 \\ r3\cdot\dfrac{v1e_n + 2\cdot\phi 2\varepsilon\cdot v2e_n + \phi 2\varepsilon^2\cdot v3e_n + q1e + r1}{v1e_n + 2\cdot\phi 2\varepsilon\cdot v2e_n + \phi 2\varepsilon^2\cdot v3e_n + q1e + r1} \end{bmatrix}$$

$k1_n := \frac{v1_n}{r1} \quad k2a_n := \frac{v2_n}{r3} \quad k2b_n := \frac{v2_n}{r1} \quad k3_n := \frac{v3_n}{r3} \qquad k1e_n := \frac{v1e_n}{r1} \quad k2ae_n := \frac{v2e_n}{r3} \quad k2be_n := \frac{v2e_n}{r1} \quad k3e_n := \frac{v3e_n}{r3}$

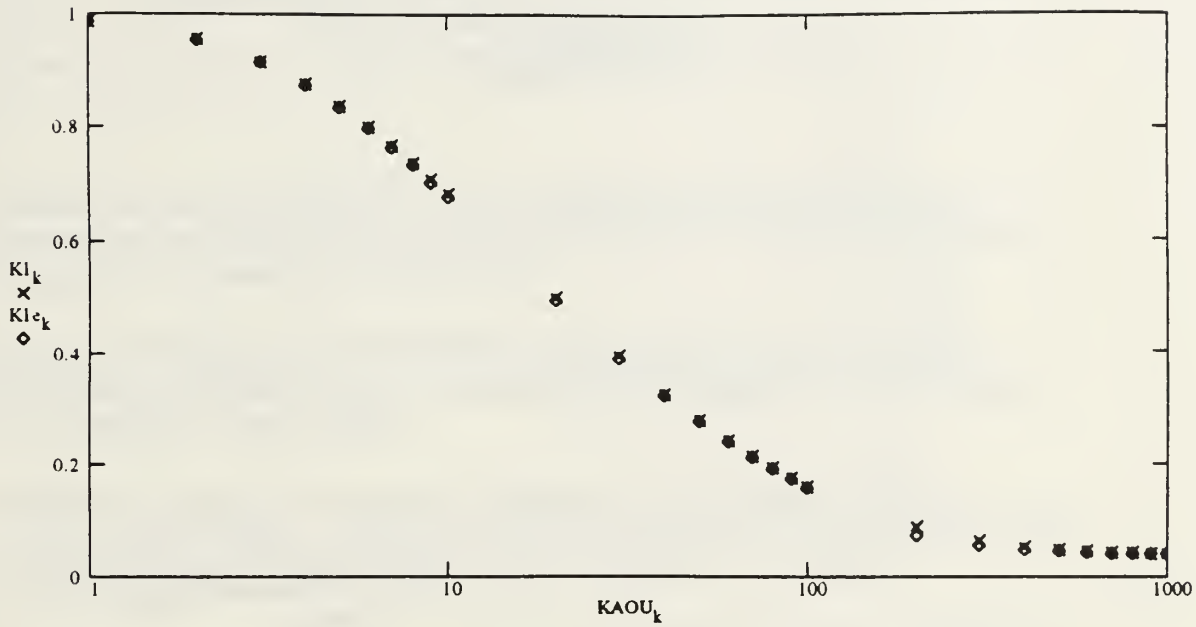
Kalman gain versus sensor AOU in nm for an interarrival time of 0.01 hours:



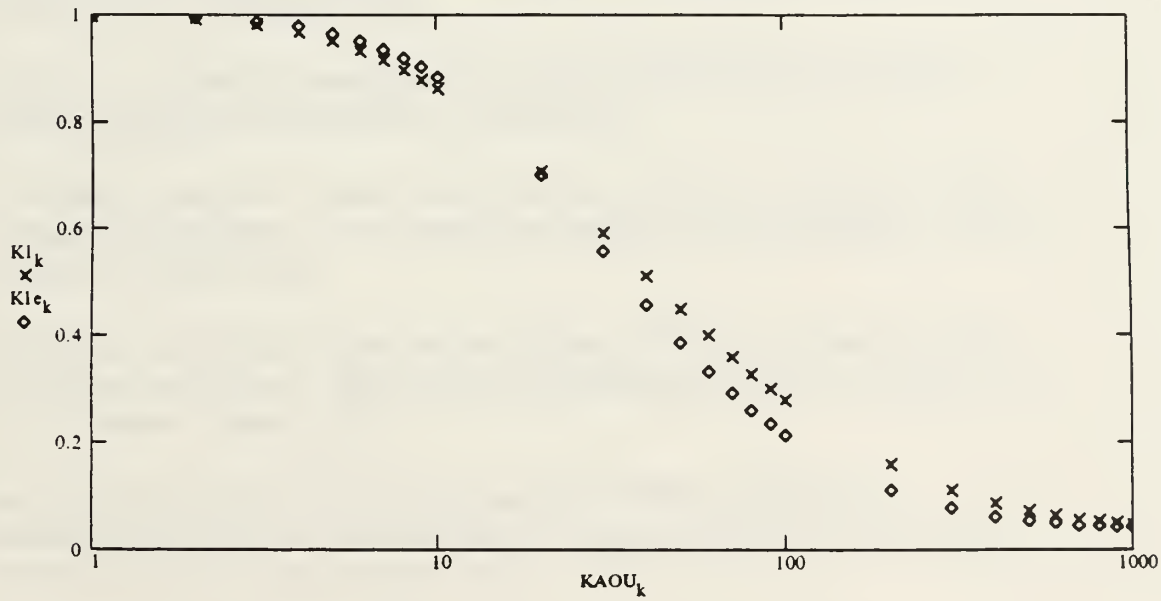
Kalman gain versus sensor AOU in nm for an interarrival time of 0.1 hours:



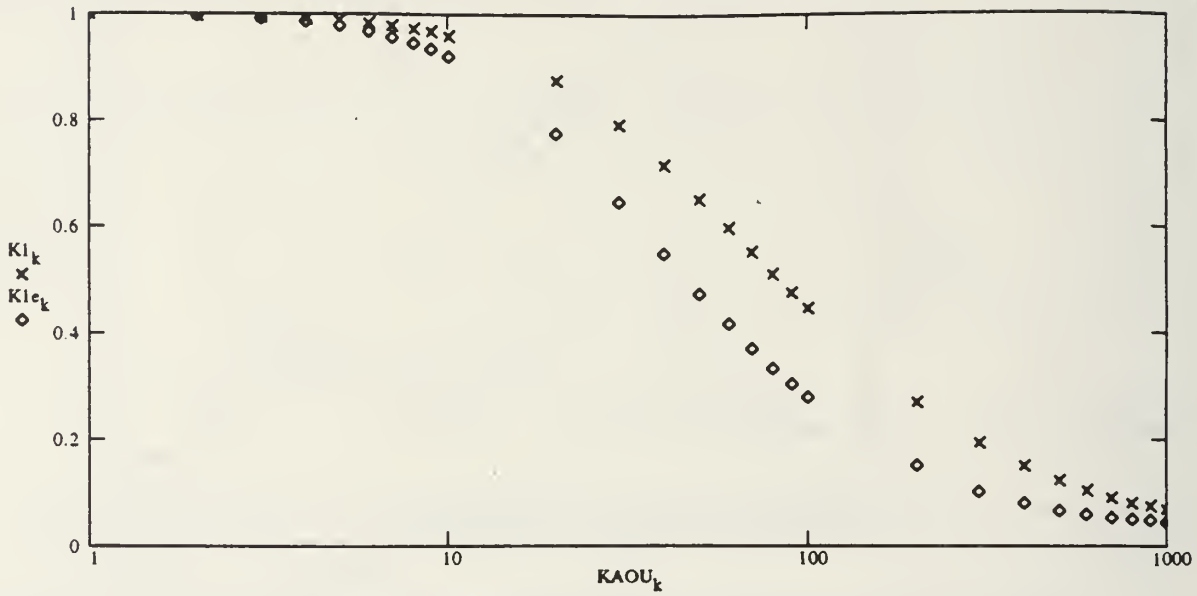
Kalman gain versus sensor AOU in nm for an interarrival time of 1 hour:



Kalman gain versus sensor AOU in nm for an interarrival time of 2 hours:



Kalman gain versus sensor AOU in nm for an interarrival time of 4 hours:



LIST OF REFERENCES

1. Metron, Inc. Report to Chief of Naval Operations (OP-71), *ASW Systems Evaluation Tool (ASSET) Technical Documentation*, by H.R. Richardson, S.C. Lent and T.A. Stefanick, 1 March 1990.

2. Metron, Inc. Report to Chief of Naval Operations (OP-71), *ASW Systems Evaluation Tool (ASSET) User's Manual*, by H.R. Richardson, S.C. Lent and T.A. Stefanick, 1 March 1990.

3. Metron, Inc. Report to Commander, Space and Naval Warfare Systems Command (SPAWAR), *A Common* LISP Implementation of the OBU Automatic Correlator Tracker, 30 May 1989.

4. TRW West Coast OBU Project Memo Number CTS.008, *Correlator Tracker Subsystem Development: Report to Track Association Design (ASSOC)*, by D.A. Robinson, 8 August, 1984.

5. Metron, Inc. Apple Macintosh Allegro Common LISP Source Code, *ASW Systems Evaluation Tool (ASSET)*, 1 June 1990.

6. TRW West Coast OBU Project Memo Number CTS.015, *Correlator Tracker Subsystem Development: One Point Track Initiation Design (START)*, by M.R. Thomsen, 26 July, 1984.

7. TRW West Coast OBU Project Memo Number CTS.015, *Correlator Tracker Subsystem Development: Multi-Point Track Initiation Design (CLUSTER)*, by M.R. Thomsen, August, 1984.

8. Naval Surface Warfare Center Technical Report NSWC-TR-88-123, *Mathematical Analysis of the Maneuvering Target Statistical Tracker (MTST) System*, by D.H. McCabe, August 1988.

9. A.R. Washburn, "Probability Density of a Moving Particle," *Operations Research*, Vol. 17, No.5, pp. 861-871, September-October 1969.

10. Daniel H. Wagner, Associates Interim Memorandum to the Applied Physics Laboratory/Johns Hopkins University, *The Ornstein-Uhlenbeck Displacement Process as a Model for Target Motion*, by B. Belkin, 1 February, 1978.

11. Sorenson, H.W., ed., *Kalman Filtering: Theory and Applications*, IEEE Press, 1985.

12. R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems,"*Trans. ASME, J. Basic Eng.*, Vol. 82D, No. 1, pp. 35-46, March 1961.

13. Wagner, D.H., "Naval Tactical Decision Aids," lecture notes taught at the Naval Postgraduate School Monterey, California, pp. III-21-III-38, Appendix B, Appendix D, September 1989.

14. Naval Undersea Systems Command Tech Report 5560, *An Adaptive Kalman Filter Tracker for Unconstrained Target Motion: Analysis and Evaluation,* by J.S. Davis and K.F. Gong, 15 June 1978.

15. Spehn, S.L., *Noise Adaptation and Correlated Maneuver Gating of an Extended Kalman Filter*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1990.

16. Naval Postgraduate School Technical Report NPS-52DE8041A, *Adaptive Tracking of Maneuvering Targets*, by J.S. Demetry and H.A. Titus, 15 April 1968.

IBUTION LIST

r

2

2

Division (OP-71B2)
3350-2000
A Rimmington 1

Metron, Inc.
11911 Freedom Drive
Suite 800
Reston VA 22090-5603 1

5. Naval Ocean System Center
Code 604
San Diego CA 92152 1

6. Professor Hal A. Titus
Code EC/Ts
Department of Electrical and Computer Engineering
Naval Postgraduate School Monterey CA 93943-5002 1

7. LT. Paul W. Vebber
419 Hillview Circle
Waukesha WI 53186 2